Sprint 2

HIGH LEVEL REQUIREMENTS

Initial user roles

User Role	**Description**
Inviter	Inviters who have a registered account with the system. These Inviters can create wishlist and register their gathering and add items to their wishlist . Also they can create invitee list and send a message. They can view the wishlist list who is buying the items for them.
ProductManager	ProductManager will add/remove the product information. They add or delete product wishlist category and check the availability of the product in inventory.
Invitee	Invitee who have free access to inviter’s wishlist . They can buy the wishlist items. Also they add shopping cart and place an order. They can view their order history and delivery status.

Initial user story descriptions

Story ID	Story description
US1	As an Invitee ,I want to sign up.
US2	As an Inviter, I want to add invitees to my gathering’s invitee’s list.
US3	As an Inviter, I want add products to my wishlist so that invitees can view and/or buy products from it.
US4	As an Invitee, I want to login into my account .
US5	As an Invitee, I want to RSVP to the gathering.
US5	As a ProductManager,I want to view all gathering’s wishlist so that I can add products accordingly.
US6	As a ProductManager ,I want to add product’s category tag so that products can be searched easily.
US7	As an Invitee, I want to view my inviter’s wishlist so that I can check the inventory for availability.
US8	As an Invitee, I want to add a product to my shopping cart from the wishlist.
US9	As an Invitee, I want to place an order from my shopping cart.
US10	As a ProductManager ,I want to add product’s category tag so that products can be searched easily.

CONCEPTUAL DESIGN

Entity: **ProductManager**
Attributes:
- username
- name[composite]
  - first_name
  - middle_name
  - last_name
- password
- address [composite]
  - address_line1
  - address_line2
  - city
  - state
  - zip_code
- email_address
- phone_number

Entity: **Product**
Attributes:
- id
- name
- description
- unit_price
- quantity

Entity: **Inviter**
Attributes:
- Username
- password
- name [composite]
  - first_name
  - middle_name
  - last_name
- phone_number

address[composite]
  address_line_1
  address_line_2
  city
  state
  Zip_code
email_address

Entity: **Gathering**
Attributes:
- id
- name
- date
- time
- description
- venue[composite]
  - address_line_1
  - address_line_2
  - city
  - state
  - zip_code

Entity: **Invitee**
Attributes:
- email_address
- password
- name [composite]
  - first_name
  - middle_name
  - last_name
- phone_number
- address[composite]
  - address_line_1
  - address_line_2
    - city
    - state
    - zip_code

Relationship: **ProductManager** adds **Product**
Cardinality: One to Many
Participation:
    ProductManager has partial participation
    Product has total participation

Relationship: **ProductManager** adds **company**
Cardinality:One to one
Participation:
    ProductManager has partial participation
    Product has total participation

Relationship:**ProductManager** adds **category**
Cardinality**:**One to many
Participation:
    Productmanager has partial participation
    Category has total participation

Relationship:**Product** has **Company**
Cardinality:many to one
Participation:
    Product has partial participation
    Category has total participation

Relationship:**Product** has **Category**
Cardinality:one to one
Participation:
    Product has partial participation
    Category has total participation

Relationship: **Inviter** creates **Gathering**
Cardinality: One to Many
Participation:
    Inviter has partial participation
    Gathering has total participation

Relationship: **Inviter** adds **Guest**
Cardinality:Many to many
Participation:

Inviter has partial participation
Guest has Total participation

Relationship:**Gathering** has **Guest**
Cardinality:Many to many
Participation:

Gathering has partial participation
Guest has total participation

Relationship:**Inviter** adds **product**
Cardinality:One to many
Participation:

Inviter has partial participation
Product has total participation

Relationship:**Gathering** has **WishlistProduct**
Cardinality:One to one
Participation:

Both will have total participation

Relationship:**Invitee** adds **product**
Cardinality:Many to many
Participation:

Invitee will have partial participation
Product will have total participation

Relationship:**Cart** has **product**
Cardinality:Many to many

Cart will have partial participation
Product will have total participation

Relationship:**Invitee** RSVP's to **Gathering**
Cardinality:Many to many

Invitee has partial participation
Gathering has partial participation

LOGICAL DESIGN

Table: **ProductManager**
Columns:

<u>Username</u>
password
first_name
middle_name
last_name
address_line1
address_line2
city
state
zipcode
email_address
phone_number

Primary key Justification: <u>username</u> will be unique for each Product Manager while signing up. So <u>username</u> becomes the primary key of the table ProductManager.

Table: **ProductCompany**
Columns:

<u>id</u>
name

Primary key Justification: id will be unique for each ProductCompany. So id becomes the primary key of the table .

Table: **ProductCategory**
Columns:
    id
    name

Primary key Justification: id will be unique for each ProductCategory. Hence, it becomes the primary key for the table **ProductCategory**.

Table: **Product**
Columns:
    id
    name
    description
    unit_price
    quantity
    company_id[foreign key;references id of ProductCompany]
    pm_username[foreign key;references username of ProductManager]
    category_id [foreign key;references id of ProductCategory]

Foreign key approach with the column pm_username.

Primary key Justification: id will be unique for each Product. Hence, it becomes the primary key for the table Product.

Foreign key justification: As username is the primary key of the table ProductManager, it can perfectly connect ProductManager table with Product table to keep a track which Product managers are adding the which products.

Foreign key justification: As id is the primary key of the table ProductCompany, it can perfectly connect ProductCompany table with the Product table to identify the company of the particular product.

Foreign key justification:As id is the primary key of the table Product*Category , it can perfectly connect* Product*Category table with the Product table to identify the category of a particular product.*

Table: **Inviter**
Columns:
    Username

password
first_name
middle_name
last_name
email_address
phone_number
address_line1
address_line2
city
state
zipcode

Primary key Justification: username will be unique for each Inviter. Hence, it becomes the primary key for the table Inviter.

Table: **Gathering**
Columns:
Id
name
date
description
address_line1
address_line2
city
state
zip_code
inviter_username[foreign key;references username of **Inviter**]

Primary key Justification: id will be unique for each Gathering. Hence, it becomes the primary key for the table Gathering.

Foreign key justification: As *username* is the primary key of the table *Inviter*, it can perfectly connect Gathering table with *Inviter* table to identify which Inviter has created the gathering. Hence, inviter_username becomes the foreign key for the table Gathering.

Table: **GatheringGuests**
Columns:

email_address[Foreign key Primary key]
gathering_id[Foreign key;references id of Gathering]

Cross Reference approach because not all guests are signed up as invitees.

Primary key Justification: email will be unique for each person. Hence, it becomes the primary key for the table Guests.

*Foreign key justification:*As every gathering has it's own guests associating gathering with it's id is the best way to connect guests to a particular gathering.

Table: **Invitee**
Columns:
email_address
password
first_name
middle_name
last_name
phone_number
address_line1
address_line2
city
state
zipcode
Primary key Justification: email_address will be unique for each Invitee and they will be added to the guests for a gathering using their email address making it easy to associate rather than having a username. Hence, it becomes the primary key for the table **Invitee**.

Table: **InviteStatus**
Columns:
id
RSVP
gathering_id[foreign key;references id of **Gathering**]
invitee_email[foreign key;references email_address of **Invitee**]

Cross Reference approach since one Invitee may be invited to multiple gathering while one gathering can have multiple invitee's with their response stored in InviteStatus making this a viable option.

Foreign key justification: As *id* is the primary key of the table *Gathering*, it can perfectly connect Gathering table with *InviteStatus* table to identify whether *Invitee* has RSVP'd to a particular gathering. Hence, gathering_id becomes the foreign key for the table **Invitee**.

Foreign key justification: As *invitee_email* is the primary key of the table *Invitee*, it can perfectly connect Invitee entity with *InviteStatus* table to identify which Invitee has RSVP'd to which gathering . Hence, inviter_username becomes the foreign key for the table **InviteStatus**.

Table: **WishlistProduct**
Columns:

id
Quantity
gathering_id[foreign key;references id of **Gathering**]
product_id[foreign key;references id of **Product**]

Primary key Justification: id will be unique for each **WishlistProduct**. Hence, it becomes the primary key for the table **WishlistProduct**.

Foreign key justification: As *id* is the primary key of the table *Gathering*, it can perfectly connect *Gathering* table with *WishlistProduct* table to identify which *Gathering* has the particular wishlistProduct. Hence, gathering_id becomes the foreign key for the table **WishlistProduct**.

Foreign key justification: As *id* is the primary key of the table *Product*, it can perfectly connect *WishlistProduct* table with *Product* table to identify which products are in there in the inventory. Hence, product_id becomes the foreign key for the table **WishlistProduct**.

Table: **Cart**

Columns:

- <u>id</u>
- quantity
- product_id[foreign key;references id of **Product**]
- invitee_email[foreign key;references email_address of **Invitee**]

*Primary key Justification: <u>id</u> will be unique for each **Cart**. Hence, it becomes the primary key for the table **Cart**.*

Foreign key justification: As id is the primary key of the Product table ,it can perfectly connect Cart table with Product table to identify products from the invitee has chosen to pick i.e Cart.

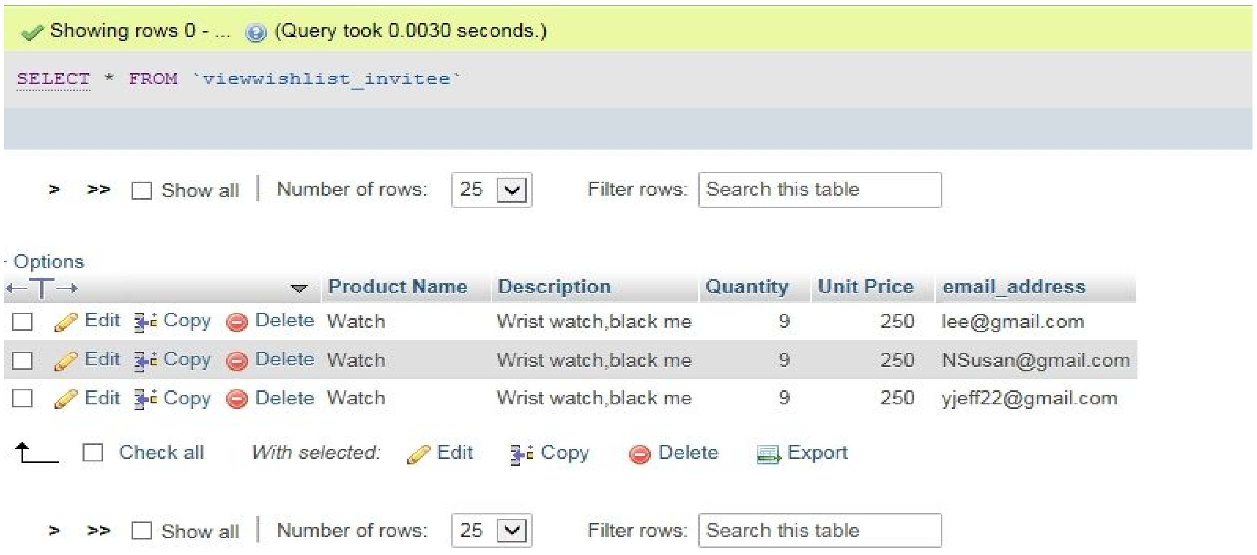
Foreign key justification:As username is the primary key of the invitee table ,it can perfectly connect Cart table with Invitee table to identify the invitee associated with that particular cart.

VIEWS AND STORED PROGRAMS

View: ViewWishlist_Invitee
Goal: The view will display the details of the products relevant to the invitees,added in the wishlist by an inviter to that particular invitee and the product manager.
The main purpose is to limit the access of a particular wishlist to the invitee of that wishlist for selection of products .

For example-

```
CREATE VIEW ViewWishlist_Invitee AS
SELECT p.name AS 'Product Name',p.description as 'Description',
   p.quantity as 'Quantity',p.unit_price AS 'Unit Price', gg.email_address
FROM gatheringguests gg
INNER JOIN wishlistproduct wl on gg.gathering_id=wl.gathering_id
INNER JOIN product p on p.id=wl.product_id
```



Showing rows 0 - ... (Query took 0.0030 seconds.)

```
SELECT * FROM `viewwishlist_invitee`
```

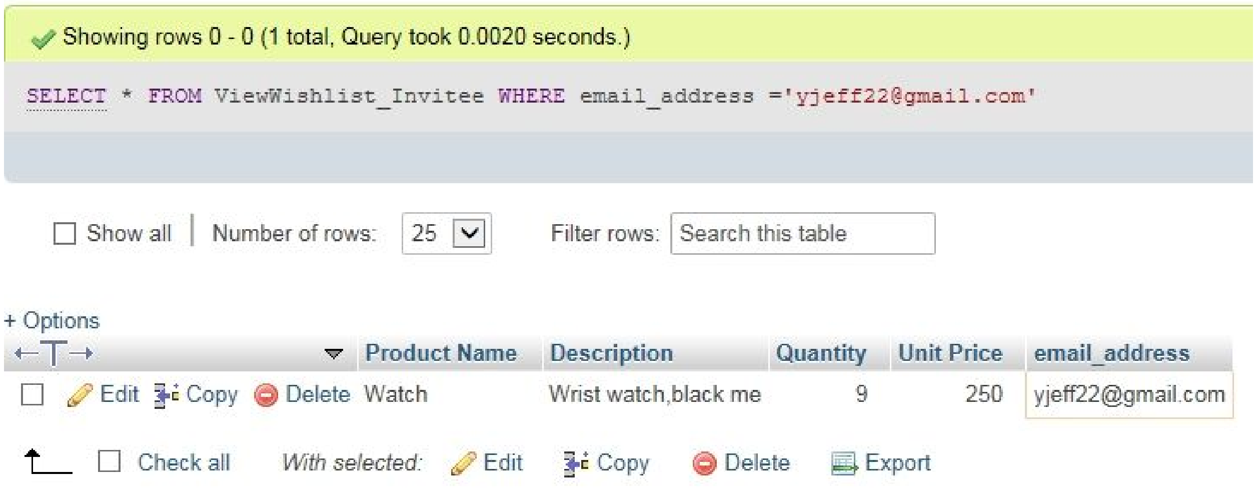
> >> Show all | Number of rows: 25 Filter rows: Search this table
Options

	Product Name	Description	Quantity	Unit Price	email_address
Edit Copy Delete	Watch	Wrist watch,black me	9	250	lee@gmail.com
Edit Copy Delete	Watch	Wrist watch,black me	9	250	NSusan@gmail.com
Edit Copy Delete	Watch	Wrist watch,black me	9	250	yjeff22@gmail.com

Check all With selected: Edit Copy Delete Export
> >> Show all | Number of rows: 25 Filter rows: Search this table

When the invitee with email address jeff22@gmail.com would view the wishlist to lookup which products to purchase the following select statement of view table.

SELECT * FROM ViewWishlist_Invitee WHERE email_address ='yjeff22@gmail.com'



Showing rows 0 - 0 (1 total, Query took 0.0020 seconds.)

```
SELECT * FROM ViewWishlist_Invitee WHERE email_address ='yjeff22@gmail.com'
```

Show all | Number of rows: 25 Filter rows: Search this table
+ Options

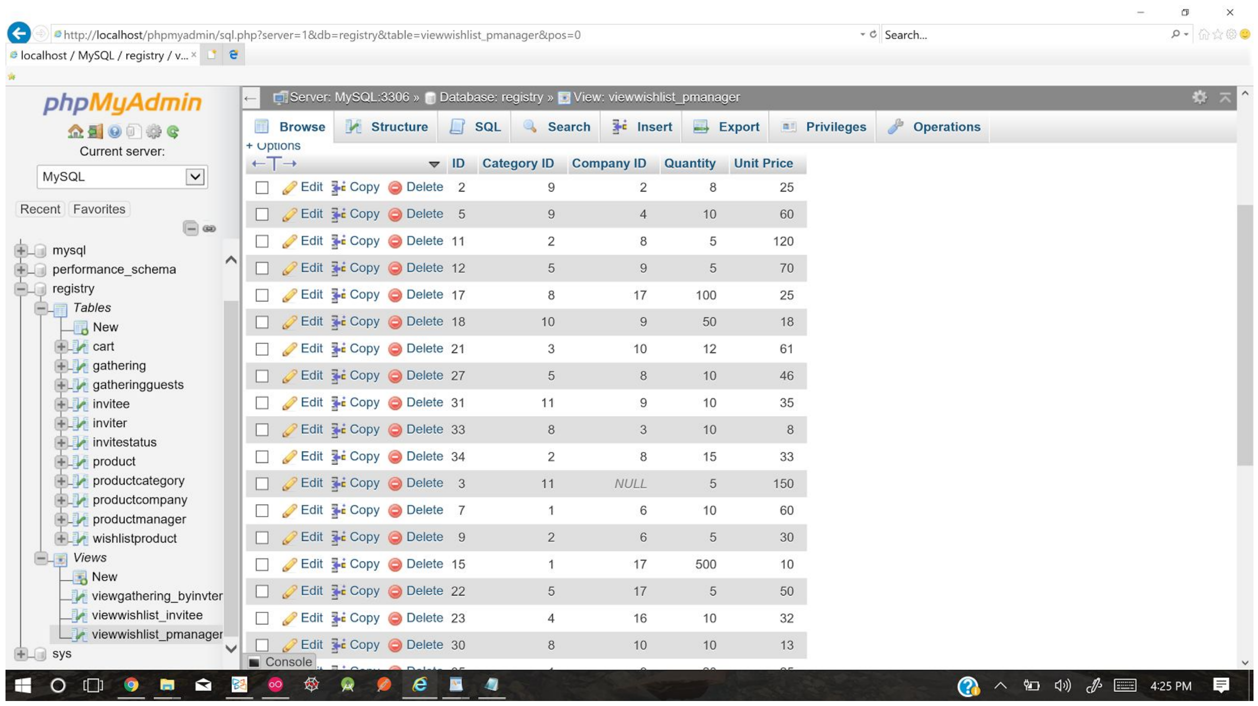
	Product Name	Description	Quantity	Unit Price	email_address
Edit Copy Delete	Watch	Wrist watch,black me	9	250	yjeff22@gmail.com

Check all With selected: Edit Copy Delete Export

View: ViewWishlist_PManager

Goal: The view will display details of the product relevant to the product manager to keep an account of the details of the products[quantity remaining,product ID etc.] to manage the inventory.

CREATE VIEW ViewWishlist_PManager AS
SELECT id AS 'ID',category_id AS 'Category ID', company_id AS 'Company ID',
  quantity AS 'Quantity',unit_price AS 'Unit Price'
FROM product p
INNER JOIN productmanager pm on p.pm_username = pm.username



http://localhost/phpmyadmin/sql.php?server=1&db=registry&table=viewwishlist_pmanager&pos=0

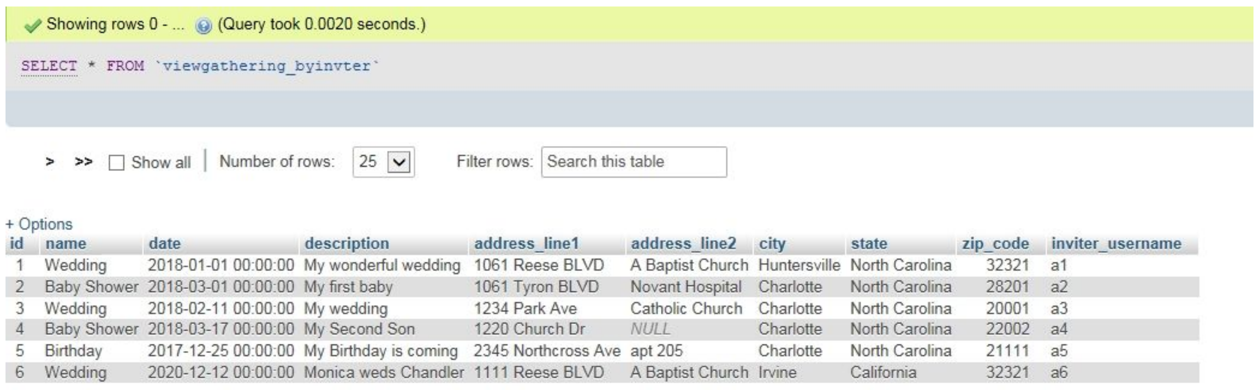
Server: MySQL:3306 » Database: registry » View: viewwishlist_pmanager

ID	Category ID	Company ID	Quantity	Unit Price
2	9	2	8	25
5	9	4	10	60
11	2	8	5	120
12	5	9	5	70
17	8	17	100	25
18	10	9	50	18
21	3	10	12	61
27	5	8	10	46
31	11	9	10	35
33	8	3	10	8
34	2	8	15	33
3	11	NULL	5	150
7	1	6	10	60
9	2	6	5	30
15	1	17	500	10
22	5	17	5	50
23	4	16	10	32
30	8	10	10	13

View: ViewGathering_byInvter

Goal: The view will display details of the gathering relevant to the inviter to keep an account of the details of the gatherings of each inviters.

CREATE VIEW ViewGathering_byInvter AS
select A.*
from gathering A
    inner join inviter B on A.inviter_username = B.username
group by A.inviter_username



Showing rows 0 - ... (Query took 0.0020 seconds.)

```
SELECT * FROM `viewgathering_byinvter`
```

> >> Show all | Number of rows: 25 Filter rows: Search this table

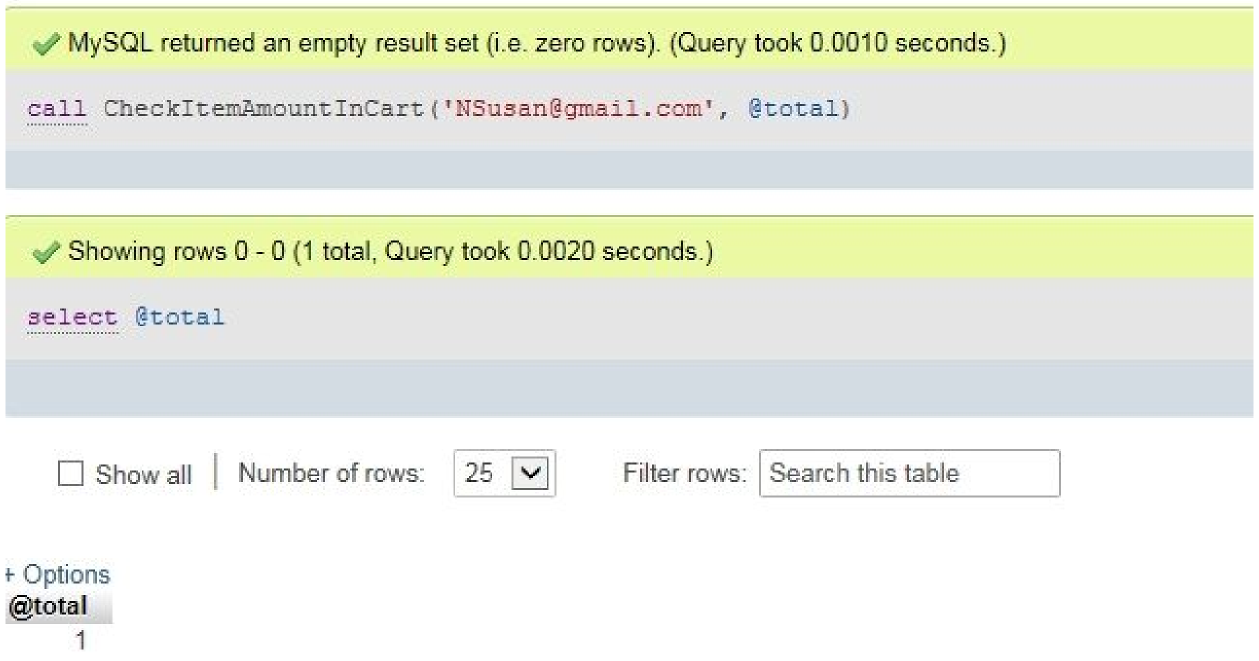
+ Options

id	name	date	description	address_line1	address_line2	city	state	zip_code	inviter_username
1	Wedding	2018-01-01 00:00:00	My wonderful wedding	1061 Reese BLVD	A Baptist Church	Huntersville	North Carolina	32321	a1
2	Baby Shower	2018-03-01 00:00:00	My first baby	1061 Tyron BLVD	Novant Hospital	Charlotte	North Carolina	28201	a2
3	Wedding	2018-02-11 00:00:00	My wedding	1234 Park Ave	Catholic Church	Charlotte	North Carolina	20001	a3
4	Baby Shower	2018-03-17 00:00:00	My Second Son	1220 Church Dr	NULL	Charlotte	North Carolina	22002	a4
5	Birthday	2017-12-25 00:00:00	My Birthday is coming	2345 Northcross Ave	apt 205	Charlotte	North Carolina	21111	a5
6	Wedding	2020-12-12 00:00:00	Monica weds Chandler	1111 Reese BLVD	A Baptist Church	Irvine	California	32321	a6

Stored procedure: <name of procedure>
Parameters: <list of parameters, specifying IN/OUT/INOUT for each>
Goal: <1-2 sentence description of what the stored procedure does>

CheckItemAmountInCart
Parameters: invitee_username(IN), amount of Items(OUT)
Goal: check amount of items in the invitee's cart

```
CREATE PROCEDURE CheckItemAmountInCart(
 IN inviteeEmailAddr VARCHAR(25),
 OUT total INT)
BEGIN
 SELECT SUM(quantity)
 INTO total
 FROM Cart
 WHERE email_address = inviteeEmailAddr;
```



```
✔ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0010 seconds.)

call CheckItemAmountInCart('NSusan@gmail.com', @total)

✔ Showing rows 0 - 0 (1 total, Query took 0.0020 seconds.)

select @total
```

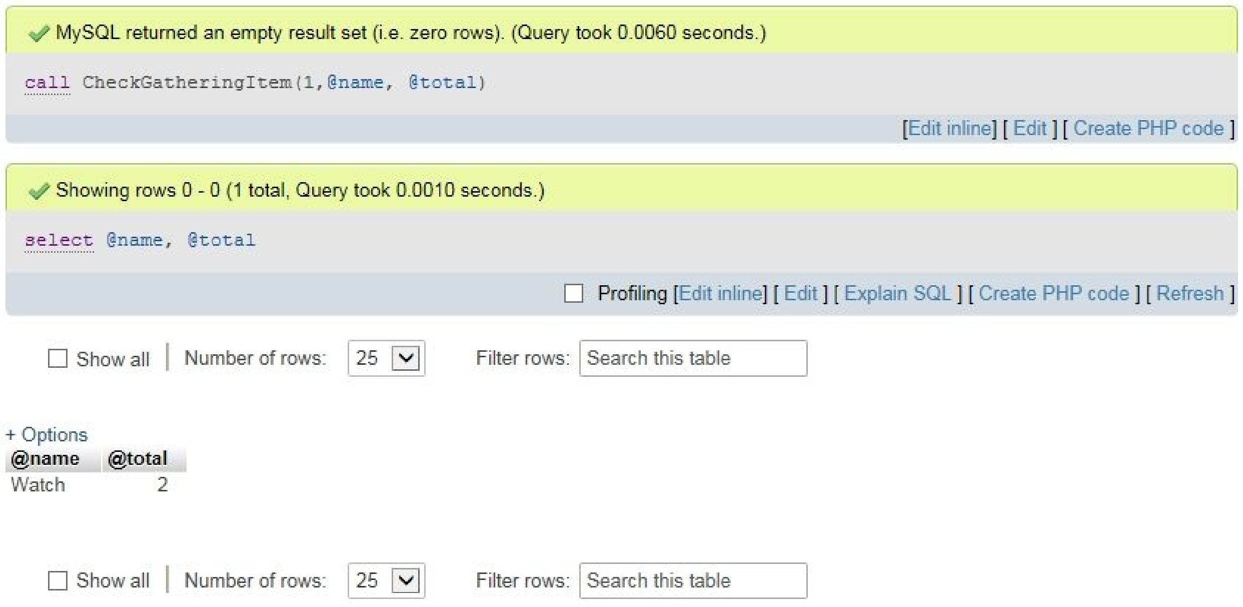
☐ Show all | Number of rows: 25 Filter rows: Search this table

+ Options

@total
1

CheckGatheringItem
Parameters: gathering_id(IN), product name(OUT), product quantity(OUT)
Goal: Output the product name and quantity registered in the gathering. The manager can figure out which items are heavily used for gatherings.

```
CREATE PROCEDURE CheckGatheringItem(
 IN gatheringId INT,
 OUT productName VARCHAR(25),
 OUT productQuantity INT)
BEGIN
 SELECT
     C.name INTO productName
  FROM
     WishlistProduct A
     INNER JOIN Product B on A.product_id = B.id
  WHERE
    A.gathering_id = gatheringId;
 SELECT
     SUM(B.quantity) INTO productQuantity
  FROM
     WishlistProduct A
     INNER JOIN Product B on A.product_id = B.id
  WHERE
    A.gathering_id = gatheringId;
END
```



```
MySQL returned an empty result set (i.e. zero rows). (Query took 0.0060 seconds.)
call CheckGatheringItem(1,@name, @total)
```

[Edit inline] [Edit] [Create PHP code]

```
Showing rows 0 - 0 (1 total, Query took 0.0010 seconds.)
select @name, @total
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all
Number of rows: 25
Filter rows: Search this table
+ Options

```
@name   @total
Watch        2
```

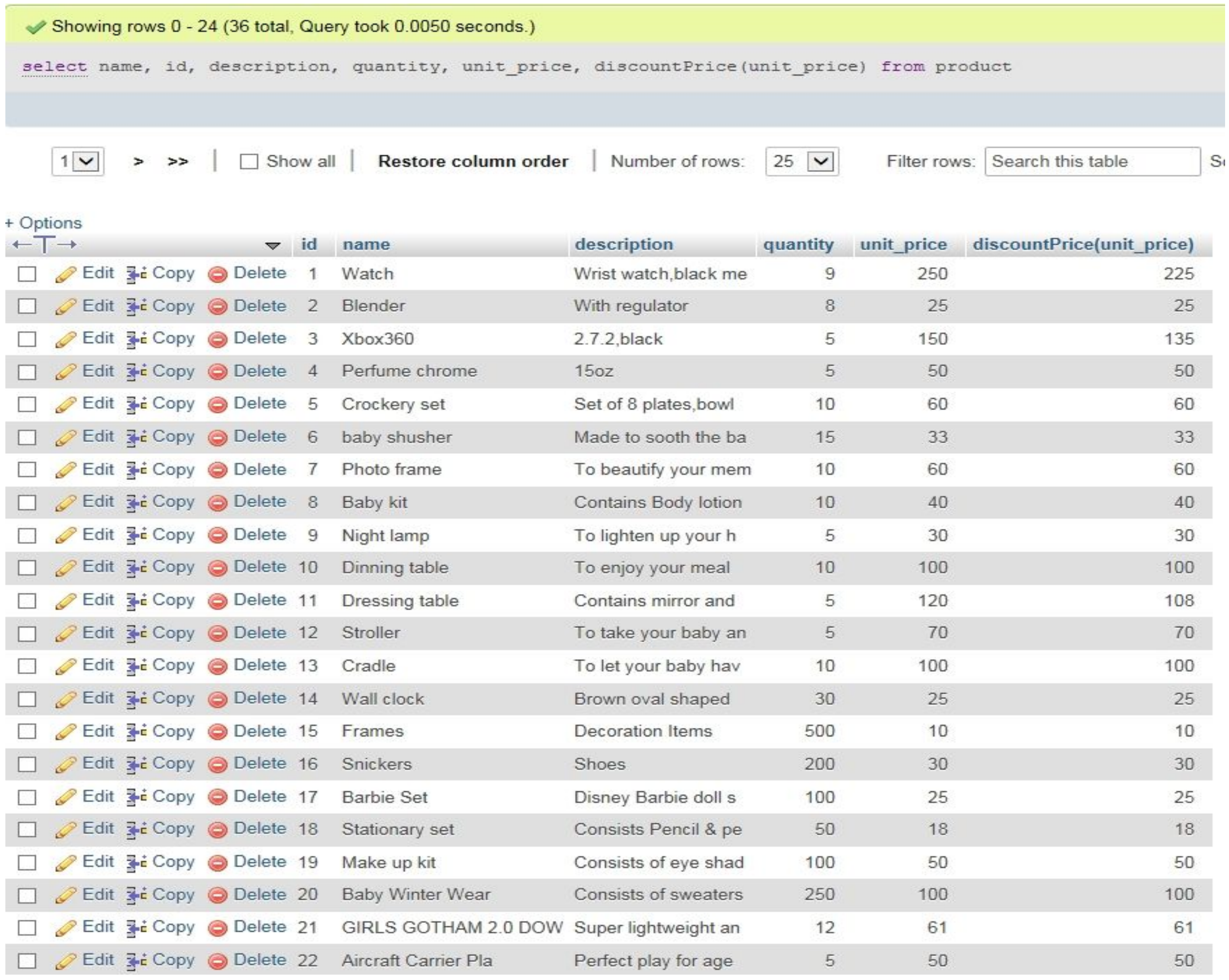
Show all
Number of rows: 25
Filter rows: Search this table

Stored function:
discountPrice
Parameters: product unit price
Goal: A function for applying a discounted amount when there is a sale event

```
CREATE FUNCTION discountPrice(price double) RETURNS double
   DETERMINISTIC
BEGIN
   DECLARE discountPrice double;

   IF price > 500 THEN
      SET discountPrice = price*0.8;
   ELSEIF price > 100 THEN
      SET discountPrice = price*0.9;
   ELSE
      SET discountPrice = price;
   END IF;

 RETURN (discountPrice);
END
```


✓ Showing rows 0 - 24 (36 total, Query took 0.0050 seconds.)

```
select name, id, description, quantity, unit_price, discountPrice(unit_price) from product
```

	id	name	description	quantity	unit_price	discountPrice(unit_price)
Edit Copy Delete	1	Watch	Wrist watch,black me	9	250	225
Edit Copy Delete	2	Blender	With regulator	8	25	25
Edit Copy Delete	3	Xbox360	2.7.2,black	5	150	135
Edit Copy Delete	4	Perfume chrome	15oz	5	50	50
Edit Copy Delete	5	Crockery set	Set of 8 plates,bowl	10	60	60
Edit Copy Delete	6	baby shusher	Made to sooth the ba	15	33	33
Edit Copy Delete	7	Photo frame	To beautify your mem	10	60	60
Edit Copy Delete	8	Baby kit	Contains Body lotion	10	40	40
Edit Copy Delete	9	Night lamp	To lighten up your h	5	30	30
Edit Copy Delete	10	Dinning table	To enjoy your meal	10	100	100
Edit Copy Delete	11	Dressing table	Contains mirror and	5	120	108
Edit Copy Delete	12	Stroller	To take your baby an	5	70	70
Edit Copy Delete	13	Cradle	To let your baby hav	10	100	100
Edit Copy Delete	14	Wall clock	Brown oval shaped	30	25	25
Edit Copy Delete	15	Frames	Decoration Items	500	10	10
Edit Copy Delete	16	Snickers	Shoes	200	30	30
Edit Copy Delete	17	Barbie Set	Disney Barbie doll s	100	25	25
Edit Copy Delete	18	Stationary set	Consists Pencil & pe	50	18	18
Edit Copy Delete	19	Make up kit	Consists of eye shad	100	50	50
Edit Copy Delete	20	Baby Winter Wear	Consists of sweaters	250	100	100
Edit Copy Delete	21	GIRLS GOTHAM 2.0 DOW	Super lightweight an	12	61	61
Edit Copy Delete	22	Aircraft Carrier Pla	Perfect play for age	5	50	50

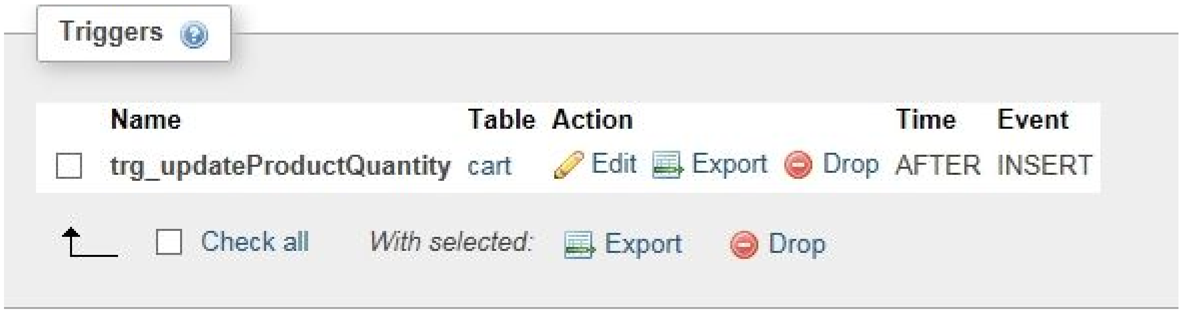
Trigger: <type of trigger> on <table name>

Goal: <1-2 sentence description of what the trigger does>

Insert trigger on Cart

Goal: If the invitee add items to their cart. It will automatically updated product table's quantity column

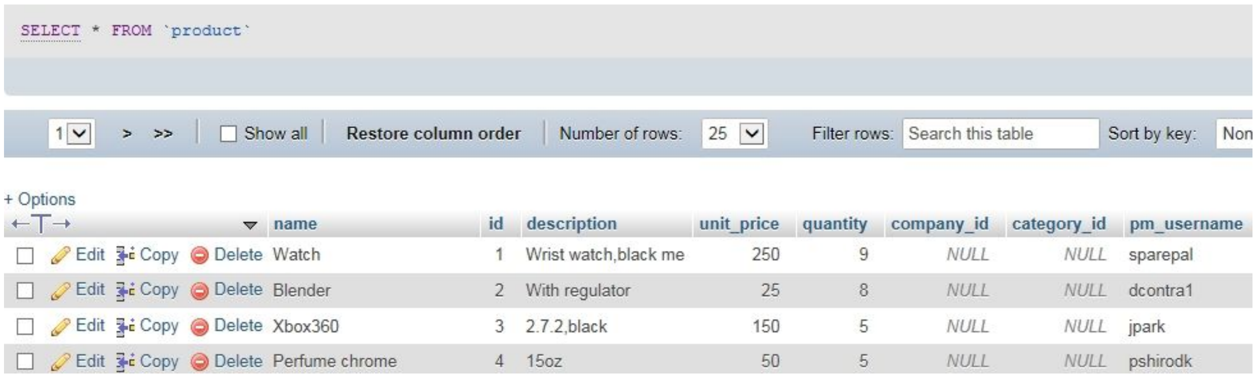
```
CREATE OR REPLACE TRIGGER trg_updateProductQuantity
AFTER INSERT ON Cart
FOR EACH ROW
BEGIN
        UPDATE product
      SET quantity = quantity - NEW.quantity
    WHERE id = ( SELECT B.product_id
              FROM cart A
                 INNER JOIN WishlistProduct B on A.product_id = B.id
              WHERE A.id = NEW.id )
END
```



Triggers

	Name	Table	Action	Time	Event
☐	trg_updateProductQuantity	cart	Edit Export Drop	AFTER	INSERT

☐ Check all *With selected:* Export Drop

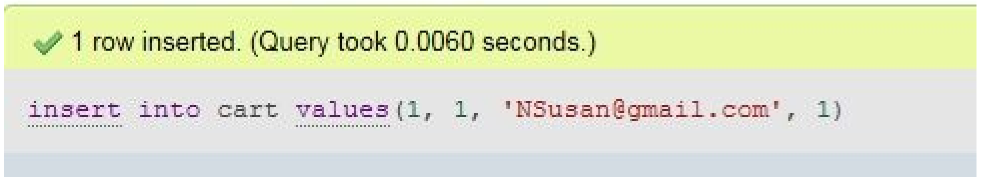


```
SELECT * FROM `product`
```

1 > >> ☐ Show all **Restore column order** Number of rows: 25 Filter rows: Search this table Sort by key: Non

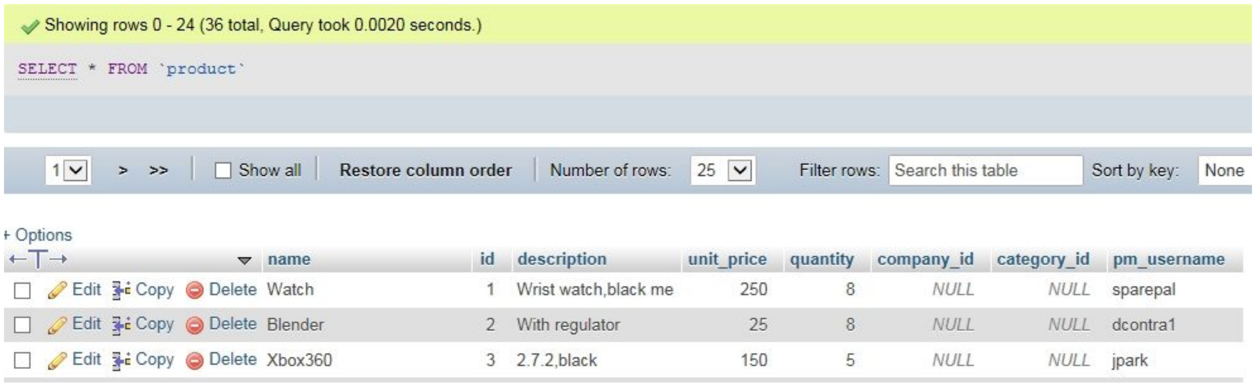
+ Options

	name	id	description	unit_price	quantity	company_id	category_id	pm_username
☐ Edit Copy Delete	Watch	1	Wrist watch,black me	250	9	*NULL*	*NULL*	sparepal
☐ Edit Copy Delete	Blender	2	With regulator	25	8	*NULL*	*NULL*	dcontra1
☐ Edit Copy Delete	Xbox360	3	2.7.2,black	150	5	*NULL*	*NULL*	jpark
☐ Edit Copy Delete	Perfume chrome	4	15oz	50	5	*NULL*	*NULL*	pshirodk



✓ 1 row inserted. (Query took 0.0060 seconds.)

```
insert into cart values(1, 1, 'NSusan@gmail.com', 1)
```



✓ Showing rows 0 - 24 (36 total, Query took 0.0020 seconds.)

```
SELECT * FROM `product`
```

1 > >> ☐ Show all **Restore column order** Number of rows: 25 Filter rows: Search this table Sort by key: None

+ Options

	name	id	description	unit_price	quantity	company_id	category_id	pm_username
☐ Edit Copy Delete	Watch	1	Wrist watch,black me	250	8	*NULL*	*NULL*	sparepal
☐ Edit Copy Delete	Blender	2	With regulator	25	8	*NULL*	*NULL*	dcontra1
☐ Edit Copy Delete	Xbox360	3	2.7.2,black	150	5	*NULL*	*NULL*	jpark