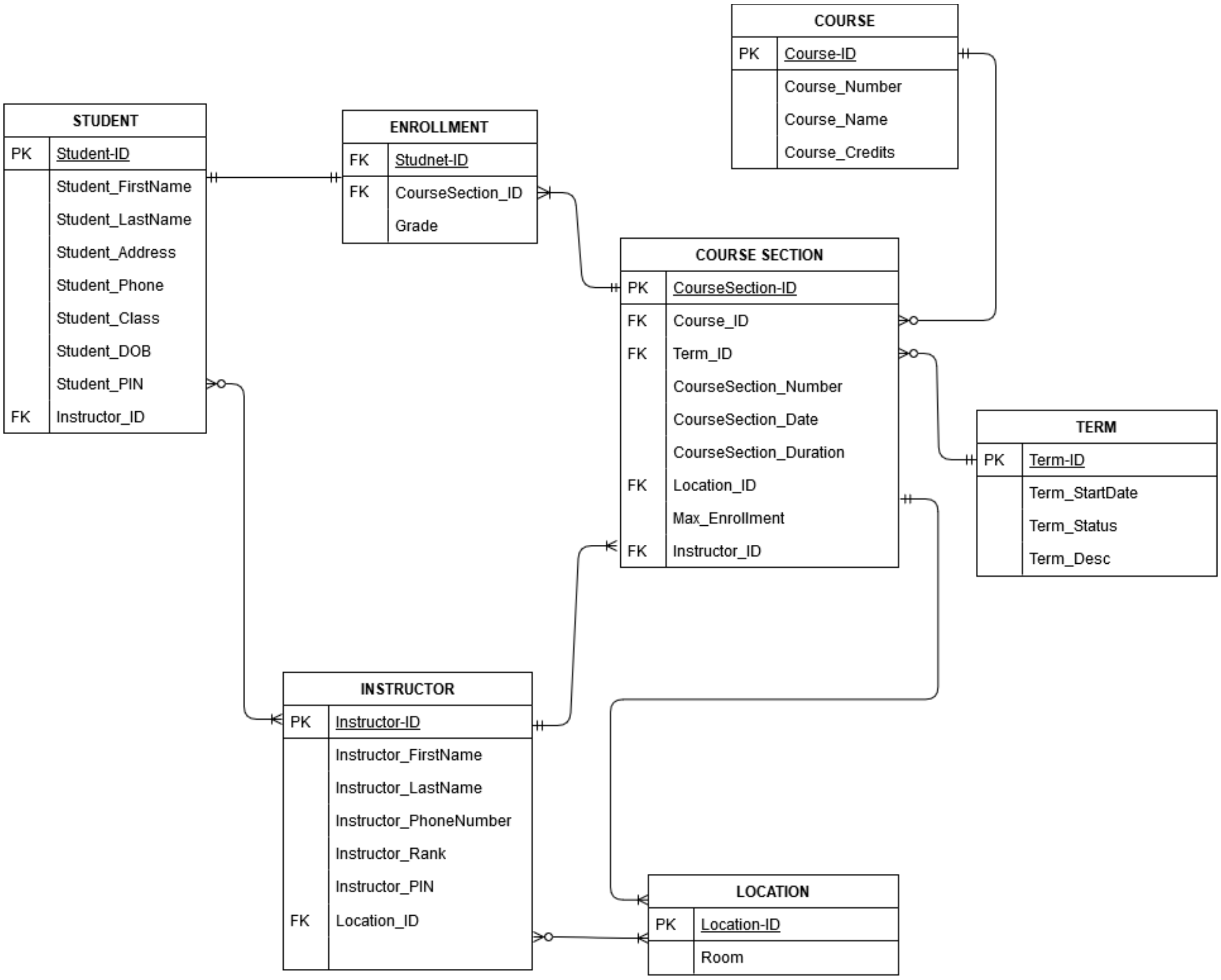
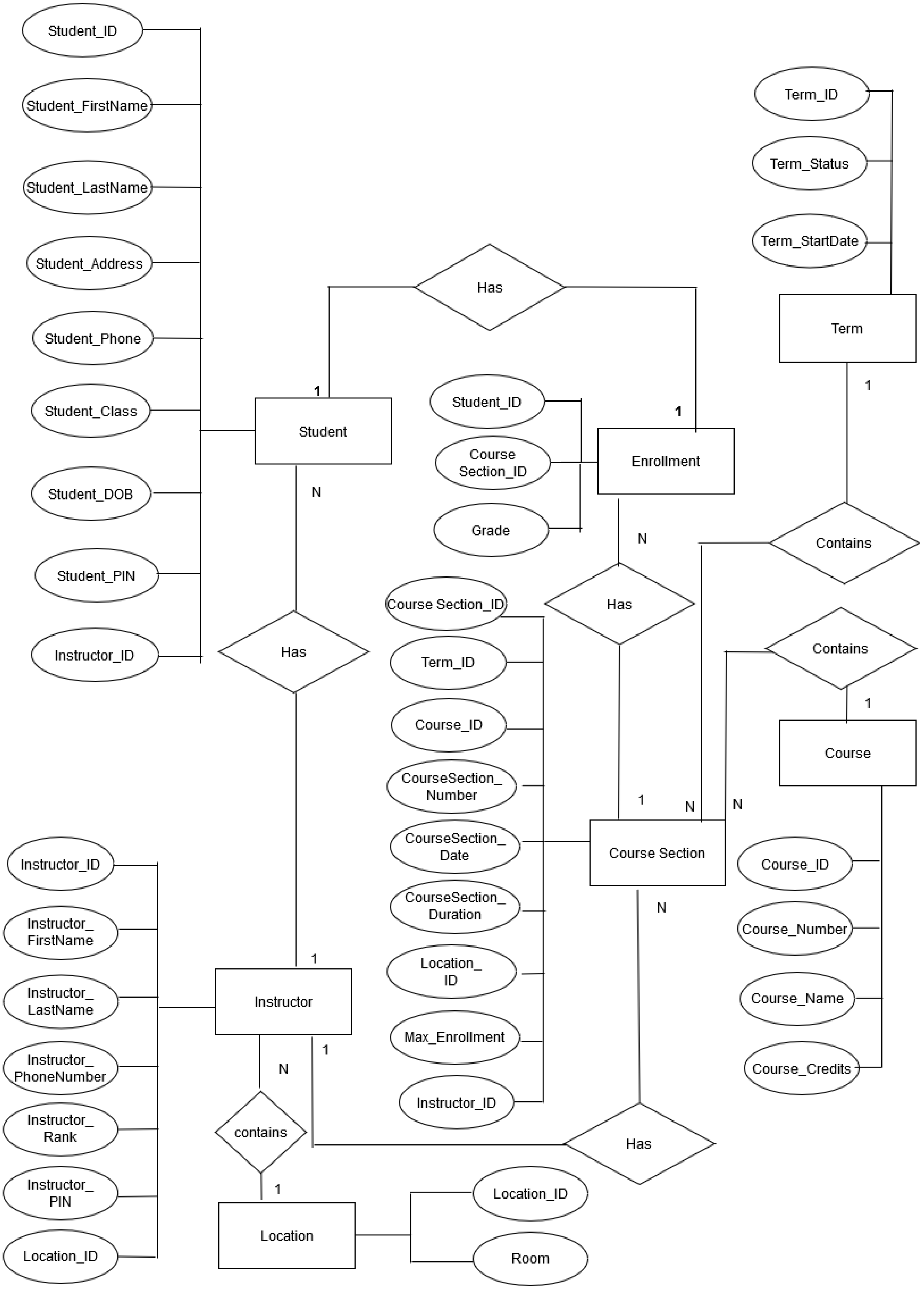
Project Three-Part 1: Northwoods University Design –

- Table Diagram using the Crow's Foot Model

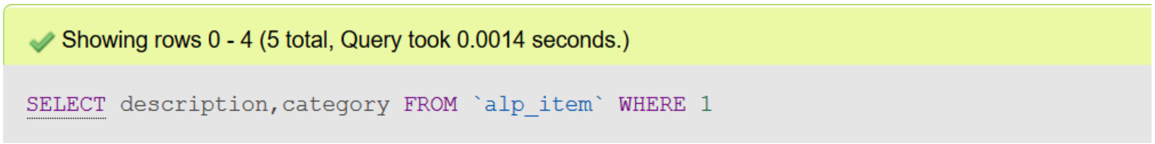
- Entity Relationship Diagram (ERD) using the Chen model




Student_ID
Student_FirstName
Student_LastName
Student_Address
Student_Phone
Student_Class
Student_DOB
Student_PIN
Instructor_ID
Student
1
N
Has
1
Enrollment
Student_ID
Course
Section_ID
Grade
N
Has
Term_ID
Term_Status
Term_StartDate
Term
1
Contains
Contains
1
Course
Has
Course Section_ID
Term_ID
Course_ID
CourseSection_
Number
CourseSection_
Date
CourseSection_
Duration
Location_
ID
Max_Enrollment
Instructor_ID
1
N
N
Course Section
N
Course_ID
Course_Number
Course_Name
Course_Credits
Instructor_ID
Instructor_
FirstName
Instructor_
LastName
Instructor_
PhoneNumber
Instructor_
Rank
Instructor_
PIN
Location_ID
1
Instructor
N
1
contains
Has
1
Location
Location_ID
Room

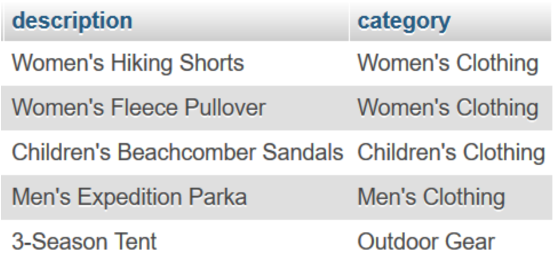
Project three Part2: Simple Queries

1. Select all the rows from the alp_item table. Display Description and Category



✓ Showing rows 0 - 4 (5 total, Query took 0.0014 seconds.)

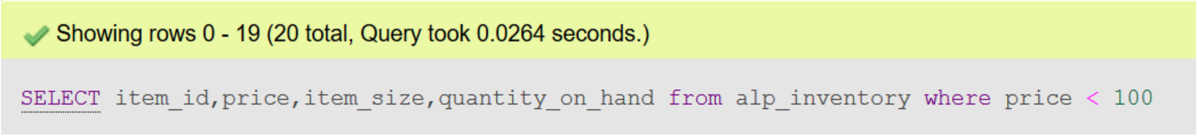
```
SELECT description,category FROM `alp_item` WHERE 1
```



description	category
Women's Hiking Shorts	Women's Clothing
Women's Fleece Pullover	Women's Clothing
Children's Beachcomber Sandals	Children's Clothing
Men's Expedition Parka	Men's Clothing
3-Season Tent	Outdoor Gear

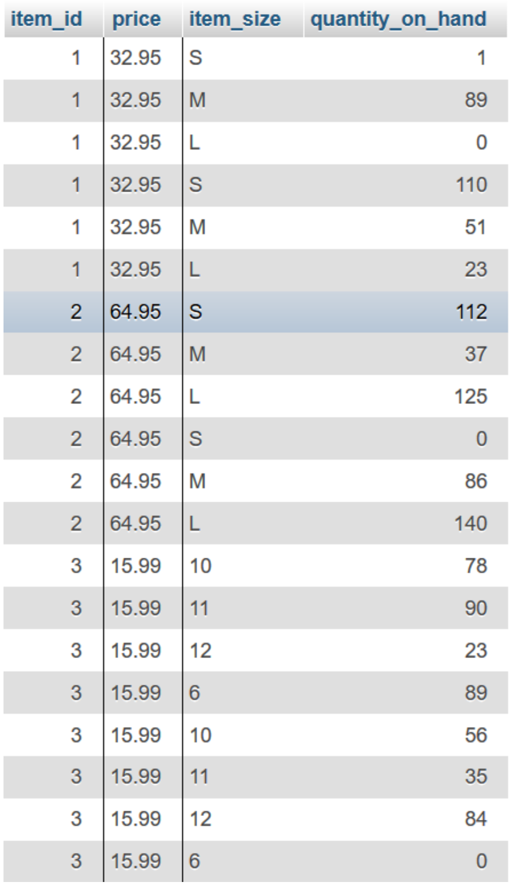
2. Select alp_inventory items that have a price of less than 100 dollars. Display Id, Size, Price and Quantity on Hand.

SELECT item_id,price,item_size,quantity_on_hand from alp_inventory where price < 100



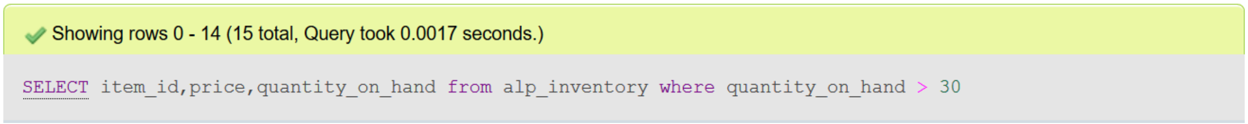
✓ Showing rows 0 - 19 (20 total, Query took 0.0264 seconds.)

```
SELECT item_id,price,item_size,quantity_on_hand from alp_inventory where price < 100
```



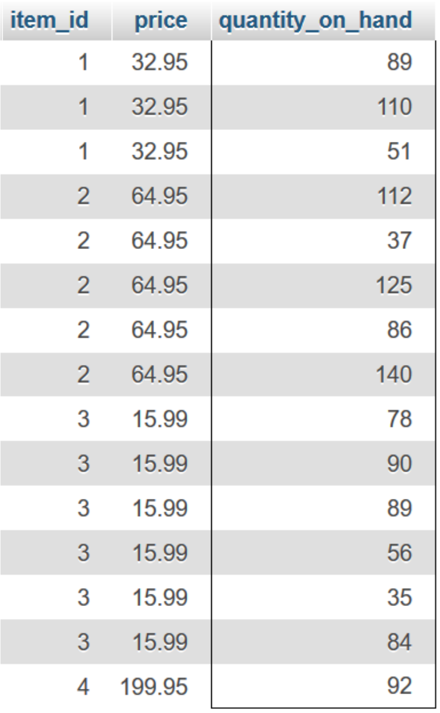
item_id	price	item_size	quantity_on_hand
1	32.95	S	1
1	32.95	M	89
1	32.95	L	0
1	32.95	S	110
1	32.95	M	51
1	32.95	L	23
2	64.95	S	112
2	64.95	M	37
2	64.95	L	125
2	64.95	S	0
2	64.95	M	86
2	64.95	L	140
3	15.99	10	78
3	15.99	11	90
3	15.99	12	23
3	15.99	6	89
3	15.99	10	56
3	15.99	11	35
3	15.99	12	84
3	15.99	6	0

3. List the alp_inventory items that have a quantity on hand of more than 30. Display Id, Quantity on Hand and Price



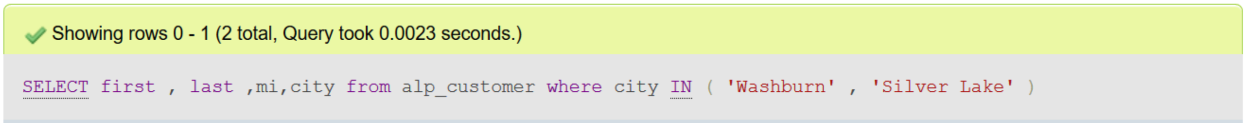
✓ Showing rows 0 - 14 (15 total, Query took 0.0017 seconds.)

```
SELECT item_id,price,quantity_on_hand from alp_inventory where quantity_on_hand > 30
```



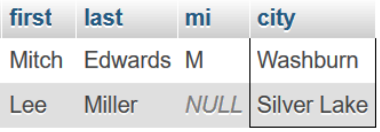
item_id	price	quantity_on_hand
1	32.95	89
1	32.95	110
1	32.95	51
2	64.95	112
2	64.95	37
2	64.95	125
2	64.95	86
2	64.95	140
3	15.99	78
3	15.99	90
3	15.99	89
3	15.99	56
3	15.99	35
3	15.99	84
4	199.95	92

4. List the alp_customers in 'Washburn ' and 'Silver Lake'. Display first_name, last_name, mi and city



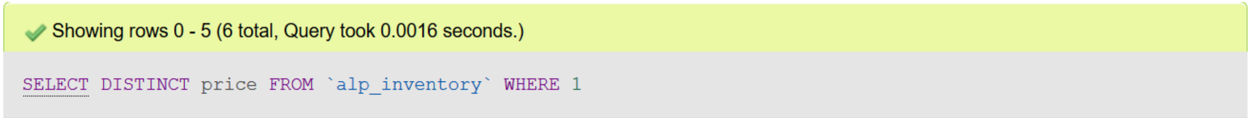
✓ Showing rows 0 - 1 (2 total, Query took 0.0023 seconds.)

```
SELECT first , last ,mi,city from alp_customer where city IN ( 'Washburn' , 'Silver Lake' )
```



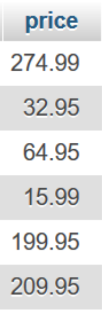
first	last	mi	city
Mitch	Edwards	M	Washburn
Lee	Miller	*NULL*	Silver Lake

5. Select the prices that occur in the alp_inventory table. A specific price should only appear once. Display the Price.



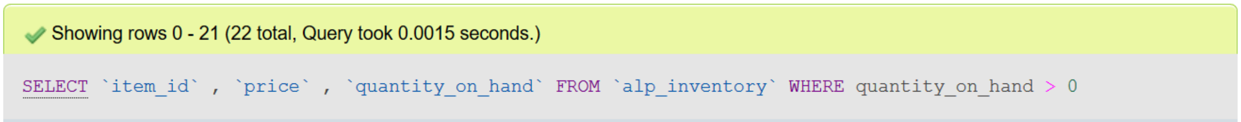
✓ Showing rows 0 - 5 (6 total, Query took 0.0016 seconds.)

```
SELECT DISTINCT price FROM `alp_inventory` WHERE 1
```



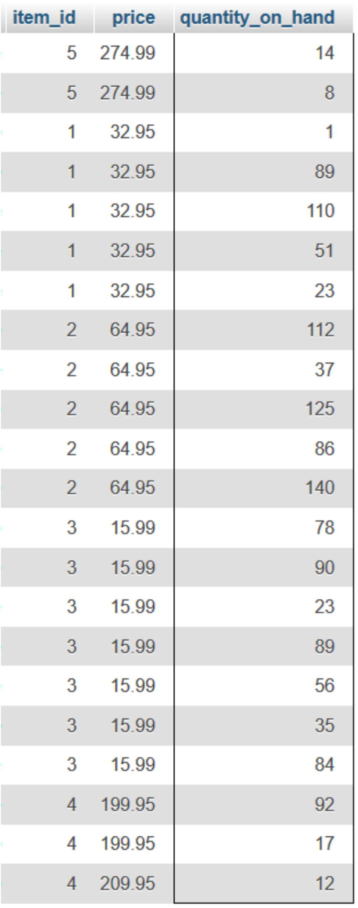
price
274.99
32.95
64.95
15.99
199.95
209.95

6. Select the alp_inventory items that are in stock. Display Id, Price and Quantity on Hand



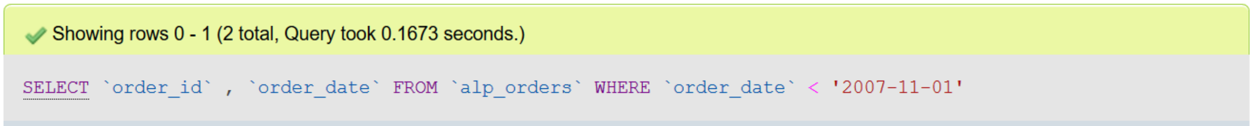
Showing rows 0 - 21 (22 total, Query took 0.0015 seconds.)

```
SELECT `item_id` , `price` , `quantity_on_hand` FROM `alp_inventory` WHERE quantity_on_hand > 0
```



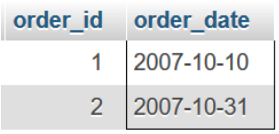
item_id	price	quantity_on_hand
5	274.99	14
5	274.99	8
1	32.95	1
1	32.95	89
1	32.95	110
1	32.95	51
1	32.95	23
2	64.95	112
2	64.95	37
2	64.95	125
2	64.95	86
2	64.95	140
3	15.99	78
3	15.99	90
3	15.99	23
3	15.99	89
3	15.99	56
3	15.99	35
3	15.99	84
4	199.95	92
4	199.95	17
4	209.95	12

7. Select the alp_orders placed before November 1, 2007. Display the Order_id and Order_date



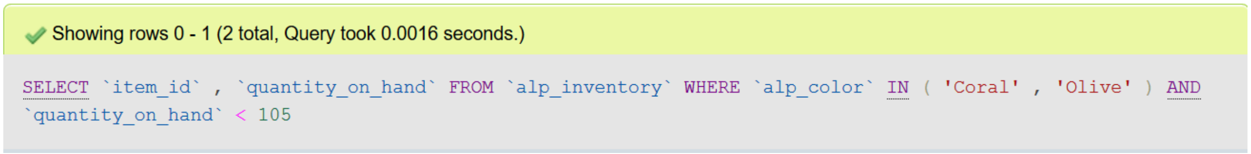
Showing rows 0 - 1 (2 total, Query took 0.1673 seconds.)

```
SELECT `order_id` , `order_date` FROM `alp_orders` WHERE `order_date` < '2007-11-01'
```



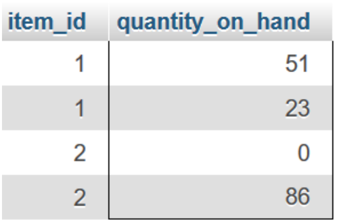
order_id	order_date
1	2007-10-10
2	2007-10-31

8. List the alp_inventory items that are ‘Coral’ or ‘Olive’ and have a Quantity on Hand of less than 105. Display Id and Quantity on Hand



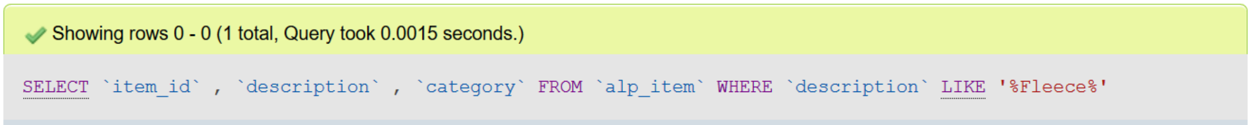
✓ Showing rows 0 - 1 (2 total, Query took 0.0016 seconds.)

```
SELECT `item_id` , `quantity_on_hand` FROM `alp_inventory` WHERE `alp_color` IN ( 'Coral' , 'Olive' ) AND
`quantity_on_hand` < 105
```



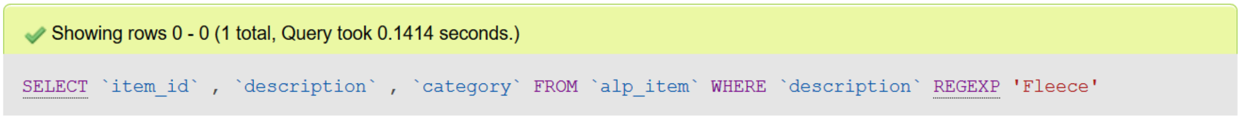
item_id	quantity_on_hand
1	51
1	23
2	0
2	86

9. List the alp_items that contain the word ‘Fleece’ in the item description. Display Id, Description, and Category



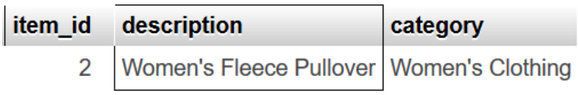
✓ Showing rows 0 - 0 (1 total, Query took 0.0015 seconds.)

```
SELECT `item_id` , `description` , `category` FROM `alp_item` WHERE `description` LIKE '%Fleece%'
```



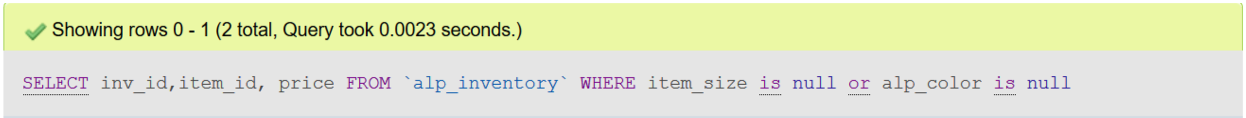
✓ Showing rows 0 - 0 (1 total, Query took 0.1414 seconds.)

```
SELECT `item_id` , `description` , `category` FROM `alp_item` WHERE `description` REGEXP 'Fleece'
```



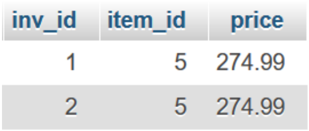
item_id	description	category
2	Women's Fleece Pullover	Women's Clothing

10. List all the alp_inventory items that do not have a size or a color assigned. Display the Id and Price.



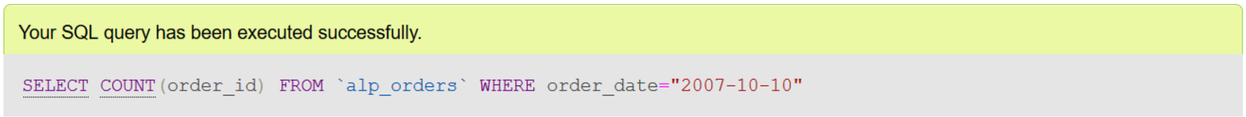
✓ Showing rows 0 - 1 (2 total, Query took 0.0023 seconds.)

```
SELECT inv_id,item_id, price FROM `alp_inventory` WHERE item_size is null or alp_color is null
```



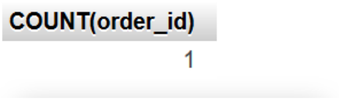
inv_id	item_id	price
1	5	274.99
2	5	274.99

11. Determine the number of orders placed on 10 October 2007. Display Number of Orders
Hint: Use the COUNT function



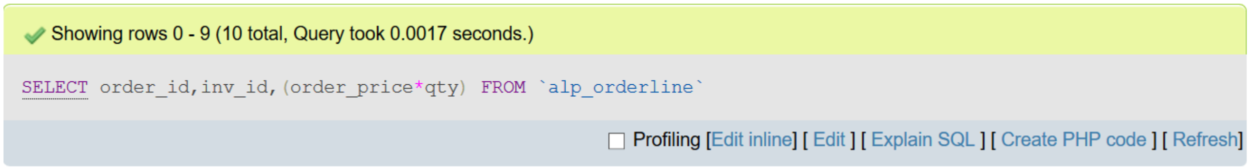
Your SQL query has been executed successfully.

```
SELECT COUNT(order_id) FROM `alp_orders` WHERE order_date="2007-10-10"
```



COUNT(order_id)
1

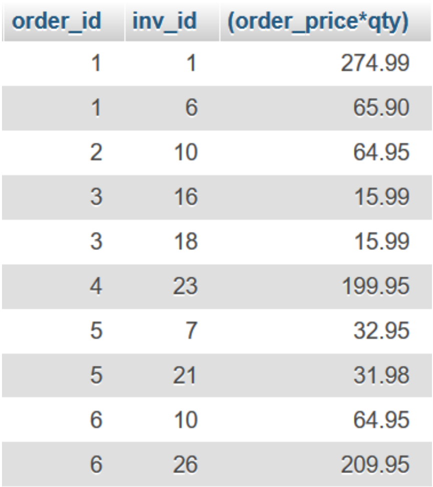
12. Determine the extended price for each row in the alp_orderline table. Display Order_id, inv_id, and Extended Price



✓ Showing rows 0 - 9 (10 total, Query took 0.0017 seconds.)

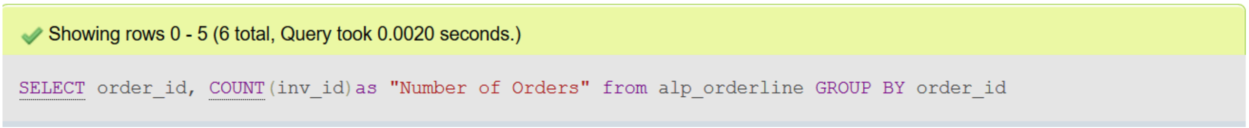
```
SELECT order_id,inv_id,(order_price*qty) FROM `alp_orderline`
```

□ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]



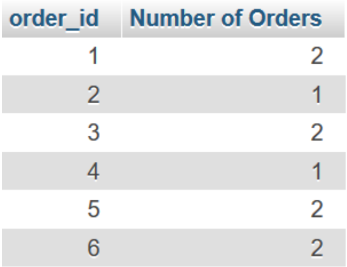
order_id	inv_id	(order_price*qty)
1	1	274.99
1	6	65.90
2	10	64.95
3	16	15.99
3	18	15.99
4	23	199.95
5	7	32.95
5	21	31.98
6	10	64.95
6	26	209.95

13. Determine the number of different items on each order. Display Order_ID and Number of Items Hint: Determine the number of different products ordered, not the total quantity ordered. This query requires a GROUP BY clause.



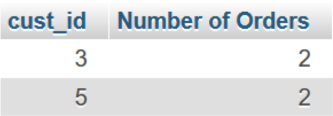
✓ Showing rows 0 - 5 (6 total, Query took 0.0020 seconds.)

```
SELECT order_id, COUNT(inv_id)as "Number of Orders" from alp_orderline GROUP BY order_id
```



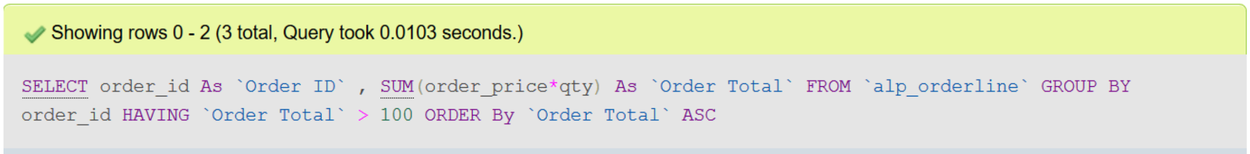
order_id	Number of Orders
1	2
2	1
3	2
4	1
5	2
6	2

14. Determine the number of orders placed by each customer. Only display the data for customers who have placed more than one order. Display Cust_id and Number of Orders Hint: This query requires a GROUP BY clause and a HAVING clause.



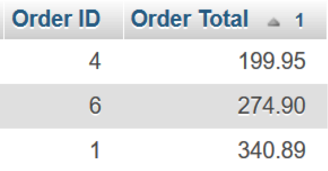
cust_id	Number of Orders
3	2
5	2

15. Determine the order total for each order that has an order total greater than 100. Display ‘*Order Id*’ and ‘*Order Total’*. Make sure the results are in ascending order total sequence.



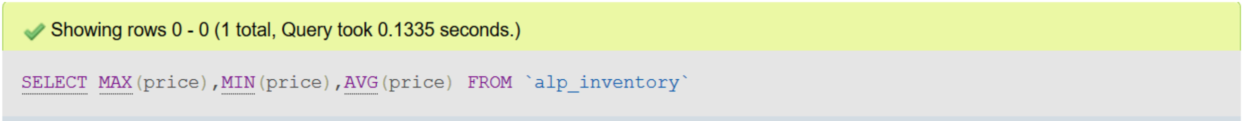
✓ Showing rows 0 - 2 (3 total, Query took 0.0103 seconds.)

```
SELECT order_id As `Order ID` , SUM(order_price*qty) As `Order Total` FROM `alp_orderline` GROUP BY
order_id HAVING `Order Total` > 100 ORDER By `Order Total` ASC
```



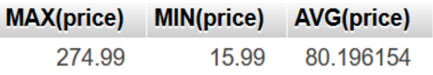
Order ID	Order Total ▲ 1
4	199.95
6	274.90
1	340.89

16. Determine what is the most expensive price, the least expensive price, and the average price in the alp_inventory table.



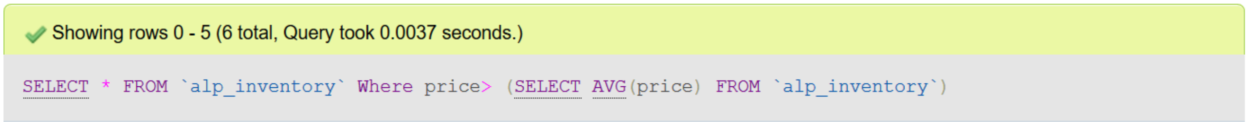
✓ Showing rows 0 - 0 (1 total, Query took 0.1335 seconds.)

```
SELECT MAX(price),MIN(price),AVG(price) FROM `alp_inventory`
```



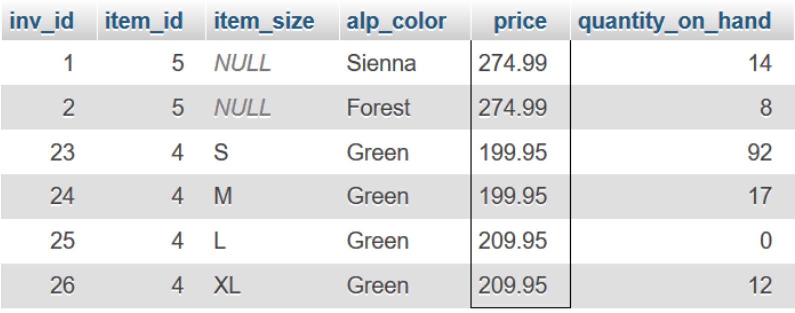
MAX(price)	MIN(price)	AVG(price)
274.99	15.99	80.196154

17. Now that you know the average price in the inventory table, display all of the information for inventory items whose price is greater than the average price.



✓ Showing rows 0 - 5 (6 total, Query took 0.0037 seconds.)

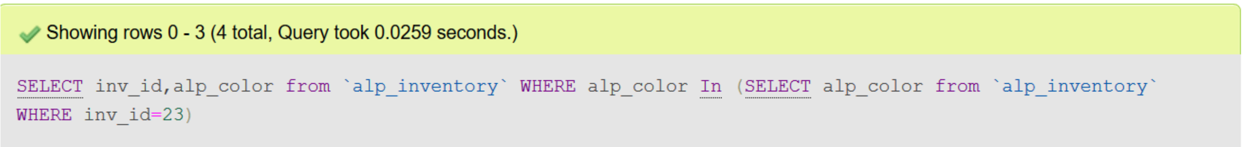
```
SELECT * FROM `alp_inventory` Where price> (SELECT AVG(price) FROM `alp_inventory`)
```



inv_id	item_id	item_size	alp_color	price	quantity_on_hand
1	5	*NULL*	Sienna	274.99	14
2	5	*NULL*	Forest	274.99	8
23	4	S	Green	199.95	92
24	4	M	Green	199.95	17
25	4	L	Green	209.95	0
26	4	XL	Green	209.95	12

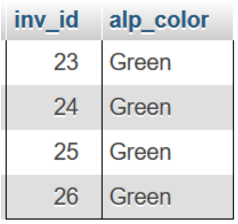
Project Three part 3: Advanced Queries

1. List the inventory items that are the same color as inventory item 23. Display the Inventory_ID and color (You must use a subquery in this query. You cannot hard-code the color.)



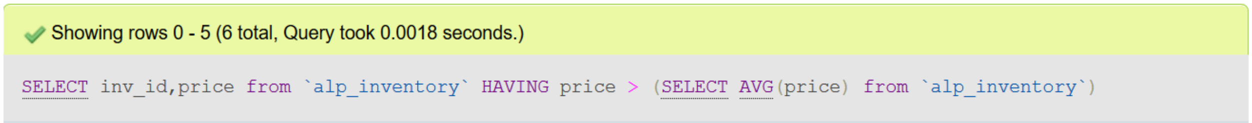
✓ Showing rows 0 - 3 (4 total, Query took 0.0259 seconds.)

```
SELECT inv_id,alp_color from `alp_inventory` WHERE alp_color In (SELECT alp_color from `alp_inventory`
WHERE inv_id=23)
```



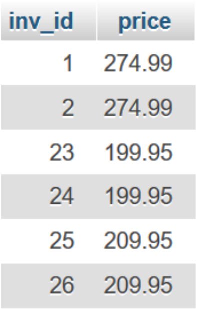
inv_id	alp_color
23	Green
24	Green
25	Green
26	Green

2. List the inventory items that have an average price greater than the average price of all the inventory items.
   Display the Inventory_ID and the Price (You must use a subquery in this query)



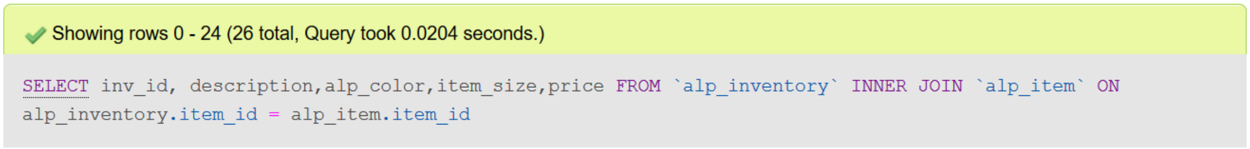
✓ Showing rows 0 - 5 (6 total, Query took 0.0018 seconds.)

```
SELECT inv_id,price from `alp_inventory` HAVING price > (SELECT AVG(price) from `alp_inventory`)
```



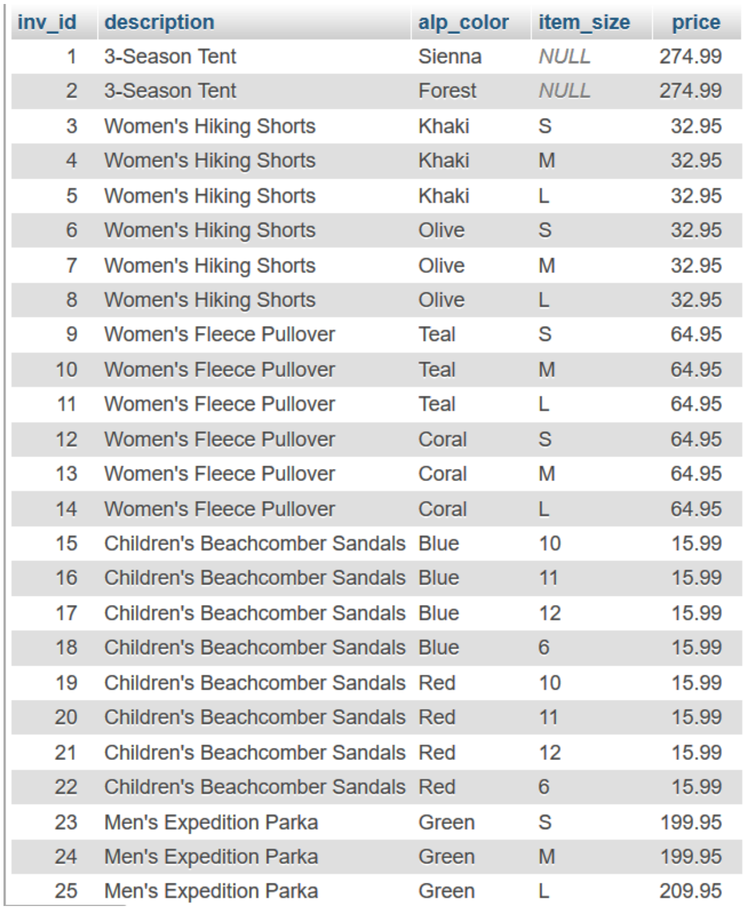
inv_id	price
1	274.99
2	274.99
23	199.95
24	199.95
25	209.95
26	209.95

3. Select the following for each inventory item: Inventory_ID, Item Description, item_size, color, and Price



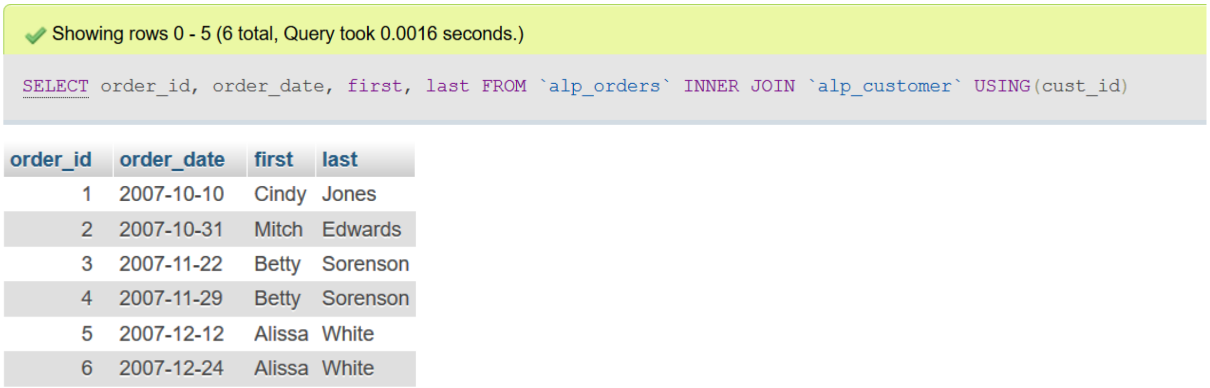
✓ Showing rows 0 - 24 (26 total, Query took 0.0204 seconds.)

```
SELECT inv_id, description,alp_color,item_size,price FROM `alp_inventory` INNER JOIN `alp_item` ON
alp_inventory.item_id = alp_item.item_id
```



inv_id	description	alp_color	item_size	price
1	3-Season Tent	Sienna	*NULL*	274.99
2	3-Season Tent	Forest	*NULL*	274.99
3	Women's Hiking Shorts	Khaki	S	32.95
4	Women's Hiking Shorts	Khaki	M	32.95
5	Women's Hiking Shorts	Khaki	L	32.95
6	Women's Hiking Shorts	Olive	S	32.95
7	Women's Hiking Shorts	Olive	M	32.95
8	Women's Hiking Shorts	Olive	L	32.95
9	Women's Fleece Pullover	Teal	S	64.95
10	Women's Fleece Pullover	Teal	M	64.95
11	Women's Fleece Pullover	Teal	L	64.95
12	Women's Fleece Pullover	Coral	S	64.95
13	Women's Fleece Pullover	Coral	M	64.95
14	Women's Fleece Pullover	Coral	L	64.95
15	Children's Beachcomber Sandals	Blue	10	15.99
16	Children's Beachcomber Sandals	Blue	11	15.99
17	Children's Beachcomber Sandals	Blue	12	15.99
18	Children's Beachcomber Sandals	Blue	6	15.99
19	Children's Beachcomber Sandals	Red	10	15.99
20	Children's Beachcomber Sandals	Red	11	15.99
21	Children's Beachcomber Sandals	Red	12	15.99
22	Children's Beachcomber Sandals	Red	6	15.99
23	Men's Expedition Parka	Green	S	199.95
24	Men's Expedition Parka	Green	M	199.95
25	Men's Expedition Parka	Green	L	209.95

4. Select all orders and the names of the customers who placed the orders. Display Order_ID, Order_Date and the Customer Name

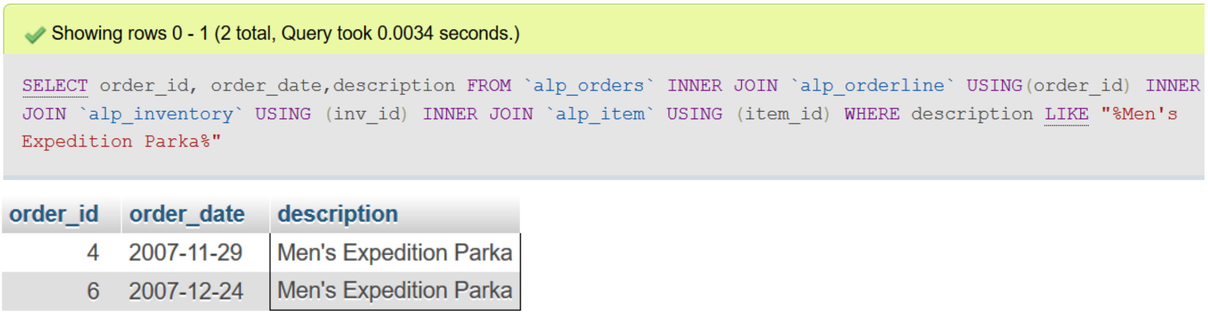


✓ Showing rows 0 - 5 (6 total, Query took 0.0016 seconds.)

```
SELECT order_id, order_date, first, last FROM `alp_orders` INNER JOIN `alp_customer` USING(cust_id)
```

order_id	order_date	first	last
1	2007-10-10	Cindy	Jones
2	2007-10-31	Mitch	Edwards
3	2007-11-22	Betty	Sorenson
4	2007-11-29	Betty	Sorenson
5	2007-12-12	Alissa	White
6	2007-12-24	Alissa	White

5. Display orders that contain the item Men's Expedition Parka. Display Order_ID, Order Date

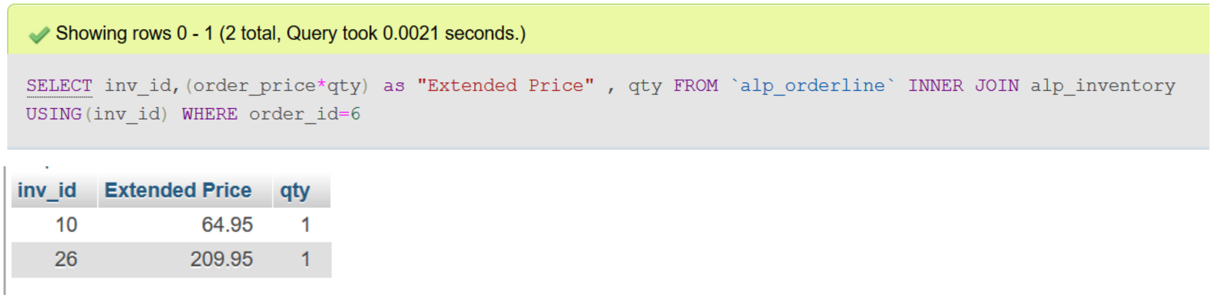


✓ Showing rows 0 - 1 (2 total, Query took 0.0034 seconds.)

```
SELECT order_id, order_date,description FROM `alp_orders` INNER JOIN `alp_orderline` USING(order_id) INNER
JOIN `alp_inventory` USING (inv_id) INNER JOIN `alp_item` USING (item_id) WHERE description LIKE "%Men's
Expedition Parka%"
```

order_id	order_date	description
4	2007-11-29	Men's Expedition Parka
6	2007-12-24	Men's Expedition Parka

6. List the items ordered for Order Id 6. Display Inventory_ID, Extended Price and Qty

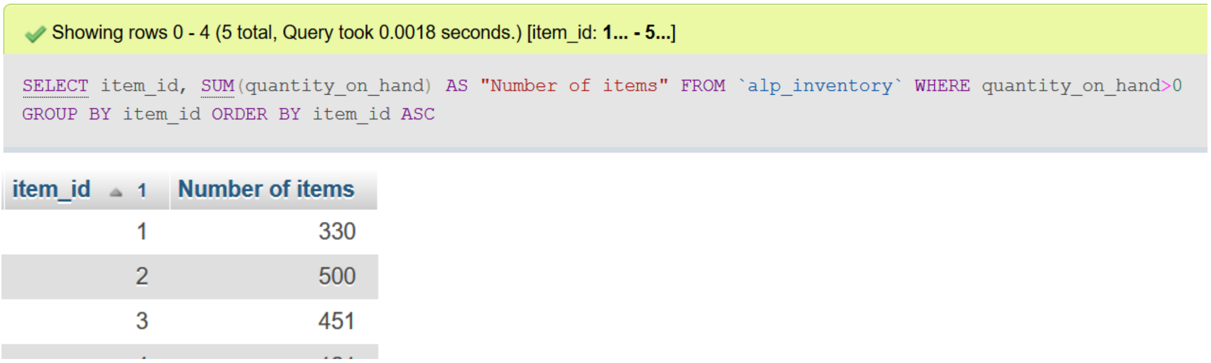


✓ Showing rows 0 - 1 (2 total, Query took 0.0021 seconds.)

```
SELECT inv_id,(order_price*qty) as "Extended Price" , qty FROM `alp_orderline` INNER JOIN alp_inventory
USING(inv_id) WHERE order_id=6
```

inv_id	Extended Price	qty
10	64.95	1
26	209.95	1

7. Display the in-stock quantity for each item. Display Item_ID and Number of items (Order the list by the Item Id in ascending order. This query requires a GROUP BY clause)

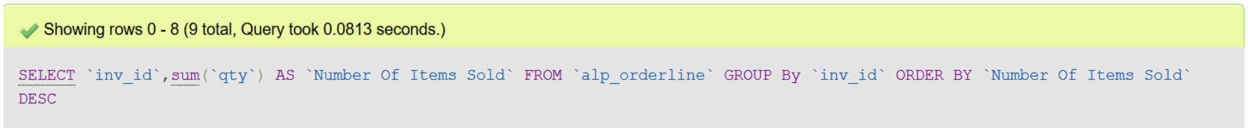


✓ Showing rows 0 - 4 (5 total, Query took 0.0018 seconds.) [item_id: **1... - 5...**]

```
SELECT item_id, SUM(quantity_on_hand) AS "Number of items" FROM `alp_inventory` WHERE quantity_on_hand>0
GROUP BY item_id ORDER BY item_id ASC
```

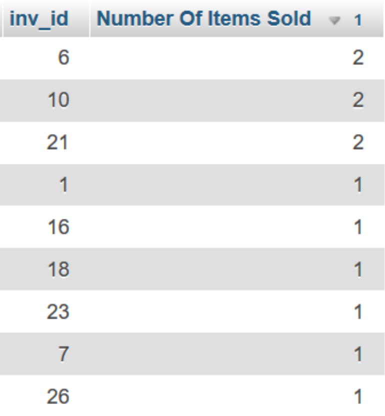
item_id ▲ 1	Number of items
1	330
2	500
3	451

8. List the quantity of each inventory item sold. Display Inventory ID and Number Sold. (Order the list by the number of items sold in descending order. This query requires a GROUP BY clause and the SUM function.)



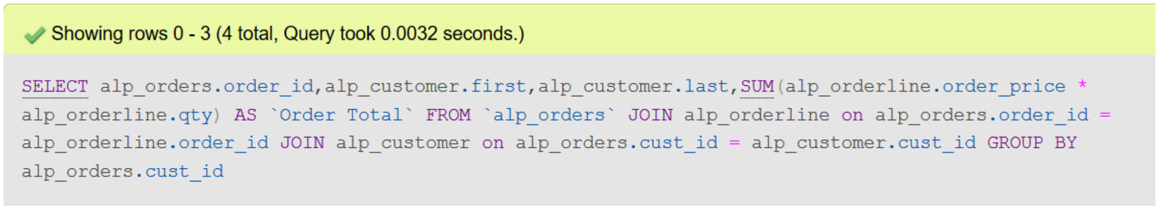
✓ Showing rows 0 - 8 (9 total, Query took 0.0813 seconds.)

```
SELECT `inv_id`,sum(`qty`) AS `Number Of Items Sold` FROM `alp_orderline` GROUP By `inv_id` ORDER BY `Number Of Items Sold`
DESC
```



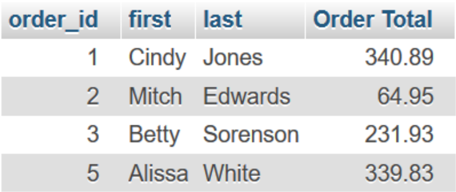
inv_id	Number Of Items Sold ▼ 1
6	2
10	2
21	2
1	1
16	1
18	1
23	1
7	1
26	1

9. Display all of the inventory information for inv_ids that do not have shipping records.



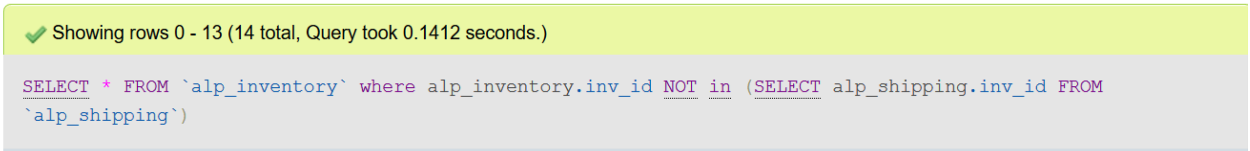
✓ Showing rows 0 - 3 (4 total, Query took 0.0032 seconds.)

```
SELECT alp_orders.order_id,alp_customer.first,alp_customer.last,SUM(alp_orderline.order_price *
alp_orderline.qty) AS `Order Total` FROM `alp_orders` JOIN alp_orderline on alp_orders.order_id =
alp_orderline.order_id JOIN alp_customer on alp_orders.cust_id = alp_customer.cust_id GROUP BY
alp_orders.cust_id
```



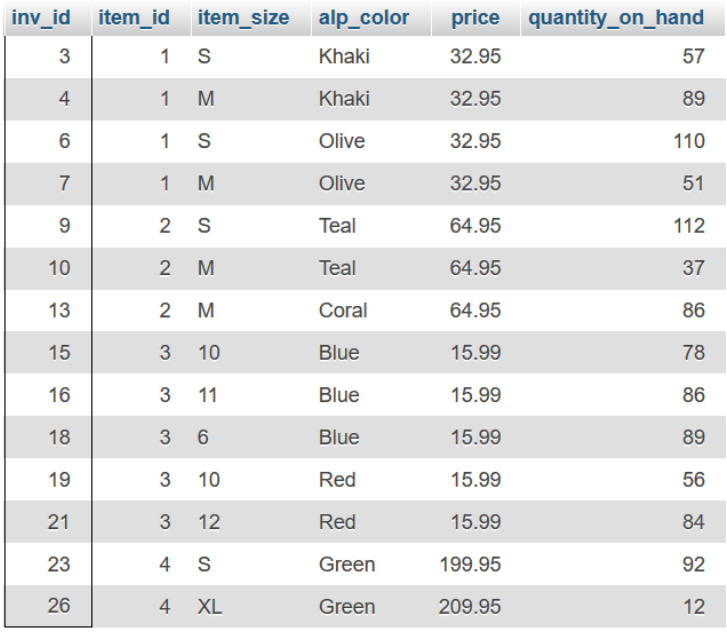
order_id	first	last	Order Total
1	Cindy	Jones	340.89
2	Mitch	Edwards	64.95
3	Betty	Sorenson	231.93
5	Alissa	White	339.83

10. Display all of the inventory information for inv_ids that do not have shipping records.



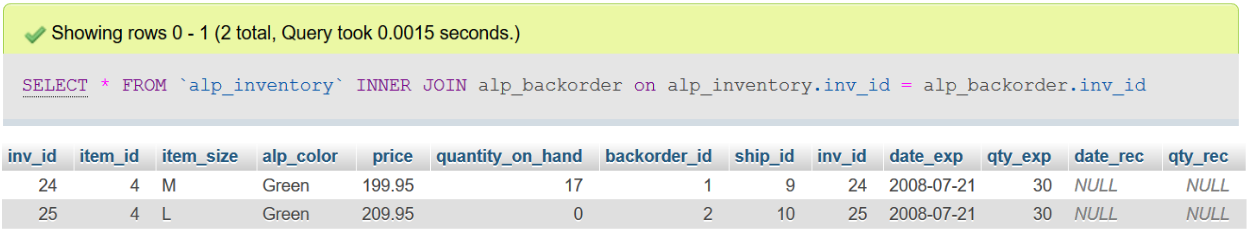
✓ Showing rows 0 - 13 (14 total, Query took 0.1412 seconds.)

```
SELECT * FROM `alp_inventory` where alp_inventory.inv_id NOT in (SELECT alp_shipping.inv_id FROM
`alp_shipping`)
```



inv_id	item_id	item_size	alp_color	price	quantity_on_hand
3	1	S	Khaki	32.95	57
4	1	M	Khaki	32.95	89
6	1	S	Olive	32.95	110
7	1	M	Olive	32.95	51
9	2	S	Teal	64.95	112
10	2	M	Teal	64.95	37
13	2	M	Coral	64.95	86
15	3	10	Blue	15.99	78
16	3	11	Blue	15.99	86
18	3	6	Blue	15.99	89
19	3	10	Red	15.99	56
21	3	12	Red	15.99	84
23	4	S	Green	199.95	92
26	4	XL	Green	209.95	12

11. Display all of the inventory information and backorder information for inv_ids that are on backorder

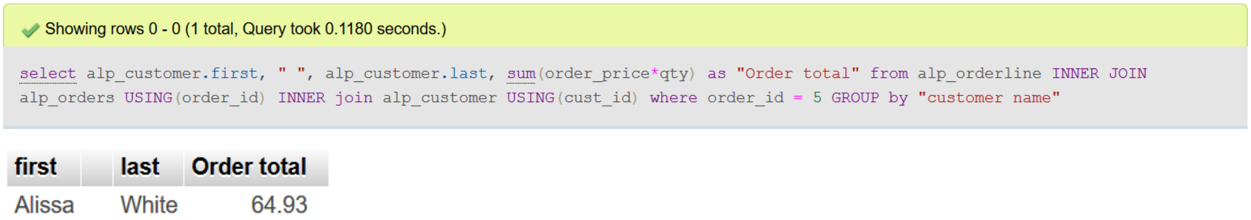


✓ Showing rows 0 - 1 (2 total, Query took 0.0015 seconds.)

```
SELECT * FROM `alp_inventory` INNER JOIN alp_backorder on alp_inventory.inv_id = alp_backorder.inv_id
```

inv_id	item_id	item_size	alp_color	price	quantity_on_hand	backorder_id	ship_id	inv_id	date_exp	qty_exp	date_rec	qty_rec
24	4	M	Green	199.95	17	1	9	24	2008-07-21	30	*NULL*	*NULL*
25	4	L	Green	209.95	0	2	10	25	2008-07-21	30	*NULL*	*NULL*

12. Display the customer first and last name and order total for order number 5

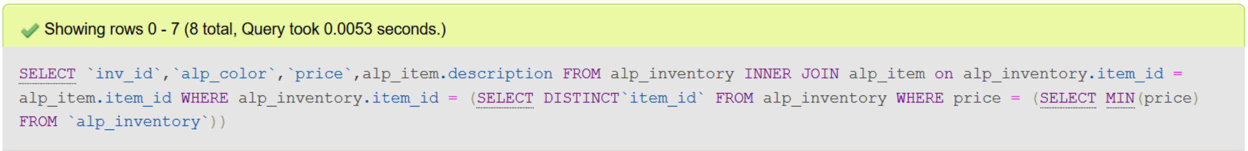


✓ Showing rows 0 - 0 (1 total, Query took 0.1180 seconds.)

```
select alp_customer.first, " ", alp_customer.last, sum(order_price*qty) as "Order total" from alp_orderline INNER JOIN
alp_orders USING(order_id) INNER join alp_customer USING(cust_id) where order_id = 5 GROUP by "customer name"
```

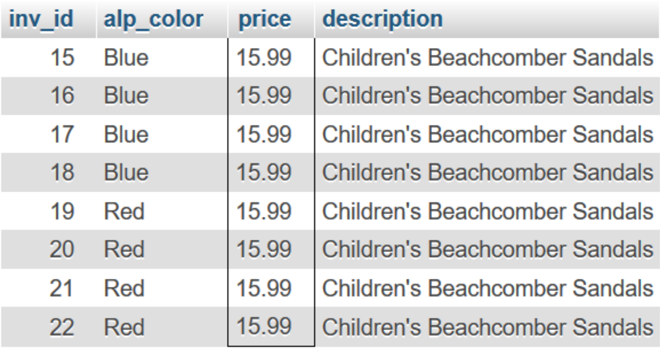
first		last	Order total
Alissa		White	64.93

13. Display the inv_id, description, price and color of the least expensive inventory item that we have in the inventory table. Use the join keyword to join the inventory table and item table. You need a subquery to determine the least expensive price then use an outer query to find all inventory items at that price.



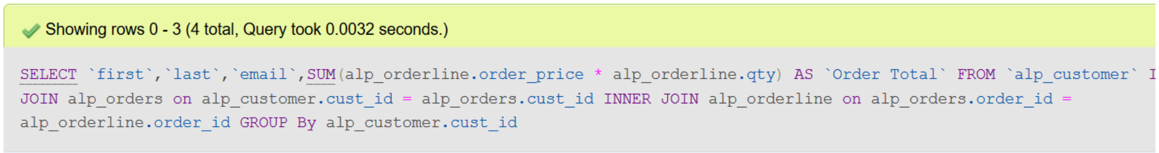
✓ Showing rows 0 - 7 (8 total, Query took 0.0053 seconds.)

```
SELECT `inv_id`,`alp_color`,`price`,alp_item.description FROM alp_inventory INNER JOIN alp_item on alp_inventory.item_id =
alp_item.item_id WHERE alp_inventory.item_id = (SELECT DISTINCT`item_id` FROM alp_inventory WHERE price = (SELECT MIN(price)
FROM `alp_inventory`))
```



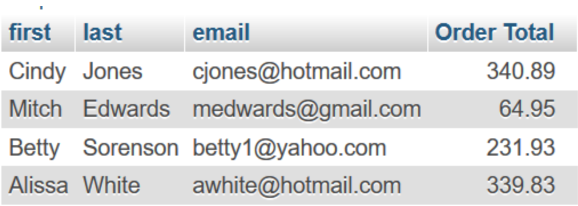
inv_id	alp_color	price	description
15	Blue	15.99	Children's Beachcomber Sandals
16	Blue	15.99	Children's Beachcomber Sandals
17	Blue	15.99	Children's Beachcomber Sandals
18	Blue	15.99	Children's Beachcomber Sandals
19	Red	15.99	Children's Beachcomber Sandals
20	Red	15.99	Children's Beachcomber Sandals
21	Red	15.99	Children's Beachcomber Sandals
22	Red	15.99	Children's Beachcomber Sandals

14. Display the first name, last name, email address and order total for customers that have placed an order at Alpine Adventures.



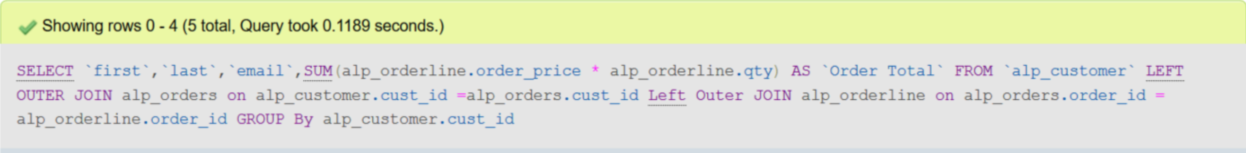
✓ Showing rows 0 - 3 (4 total, Query took 0.0032 seconds.)

```
SELECT `first`,`last`,`email`,SUM(alp_orderline.order_price * alp_orderline.qty) AS `Order Total` FROM `alp_customer` I
JOIN alp_orders on alp_customer.cust_id = alp_orders.cust_id INNER JOIN alp_orderline on alp_orders.order_id =
alp_orderline.order_id GROUP By alp_customer.cust_id
```



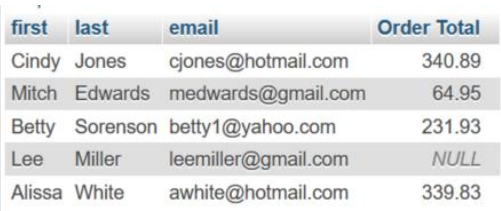
first	last	email	Order Total
Cindy	Jones	cjones@hotmail.com	340.89
Mitch	Edwards	medwards@gmail.com	64.95
Betty	Sorenson	betty1@yahoo.com	231.93
Alissa	White	awhite@hotmail.com	339.83

15. Modify the above query to display ALL customers, whether they have placed an order or not.



✓ Showing rows 0 - 4 (5 total, Query took 0.1189 seconds.)

```
SELECT `first`,`last`,`email`,SUM(alp_orderline.order_price * alp_orderline.qty) AS `Order Total` FROM `alp_customer` LEFT
OUTER JOIN alp_orders on alp_customer.cust_id =alp_orders.cust_id Left Outer JOIN alp_orderline on alp_orders.order_id =
alp_orderline.order_id GROUP By alp_customer.cust_id
```

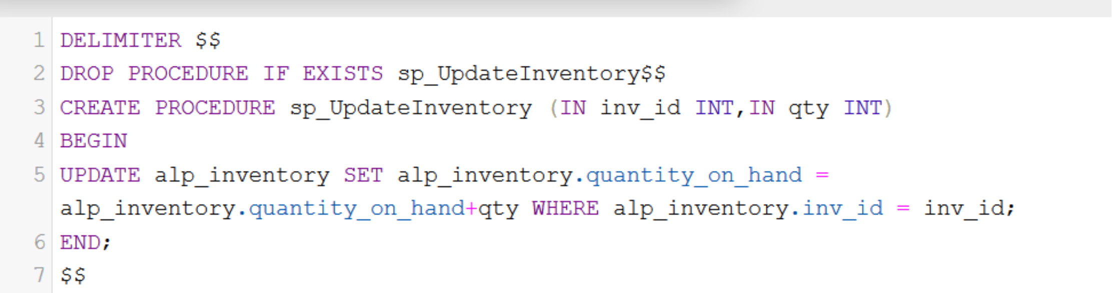


first	last	email	Order Total
Cindy	Jones	cjones@hotmail.com	340.89
Mitch	Edwards	medwards@gmail.com	64.95
Betty	Sorenson	betty1@yahoo.com	231.93
Lee	Miller	leemiller@gmail.com	NULL
Alissa	White	awhite@hotmail.com	339.83

Project three part 4 Procedures and Triggers

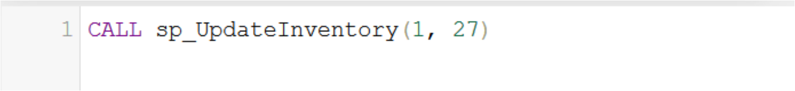
Procedure 1:

Write a procedure to update the *quantity_on_hand* column in the inventory table. It will accept two arguments (inv_id, qty). It does not return any value. The qty passed into the procedure will be added to the quantity_on_hand in the table.
Name the procedure sp_UpdateInventory



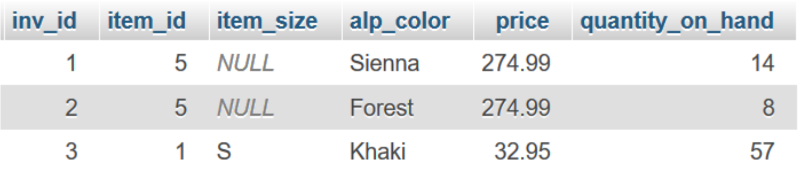
```
DELIMITER $$
DROP PROCEDURE IF EXISTS sp_UpdateInventory$$
CREATE PROCEDURE sp_UpdateInventory (IN inv_id INT,IN qty INT)
BEGIN
UPDATE alp_inventory SET alp_inventory.quantity_on_hand = alp_inventory.quantity_on_hand+qty WHERE alp_inventory.inv_id = inv_id;
END;
$$
```

Calling procedure:



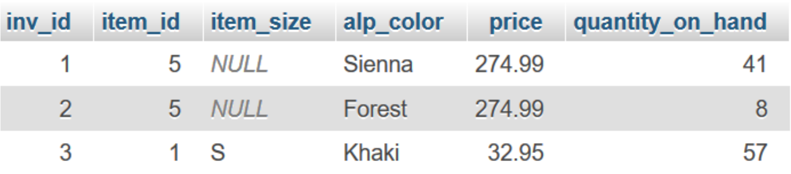
```
CALL sp_UpdateInventory(1, 27)
```

Before:



inv_id	item_id	item_size	alp_color	price	quantity_on_hand
1	5	*NULL*	Sienna	274.99	14
2	5	*NULL*	Forest	274.99	8
3	1	S	Khaki	32.95	57

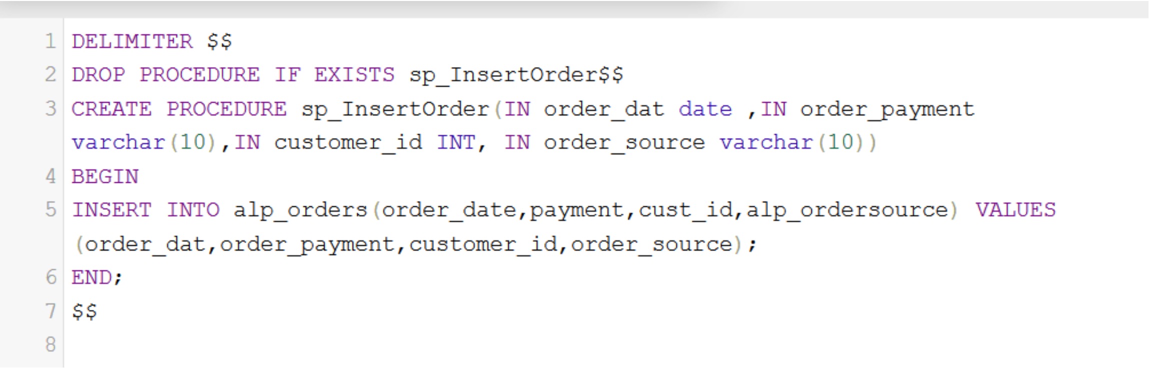
After:



inv_id	item_id	item_size	alp_color	price	quantity_on_hand
1	5	*NULL*	Sienna	274.99	41
2	5	*NULL*	Forest	274.99	8
3	1	S	Khaki	32.95	57

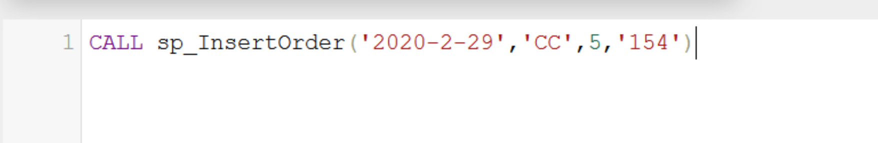
Procedure 2:

Write a procedure that will insert a row into the Orders table. It will accept arguments for each of the columns (except the order_id because that is generated by SQL Server). Name the procedure sp_InsertOrder



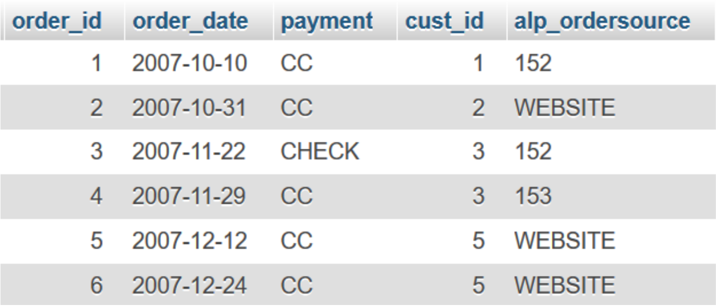
```
DELIMITER $$
DROP PROCEDURE IF EXISTS sp_InsertOrder$$
CREATE PROCEDURE sp_InsertOrder(IN order_dat date ,IN order_payment varchar(10),IN customer_id INT, IN order_source varchar(10))
BEGIN
INSERT INTO alp_orders(order_date,payment,cust_id,alp_ordersource) VALUES (order_dat,order_payment,customer_id,order_source);
END;
$$
```

Calling procedure:



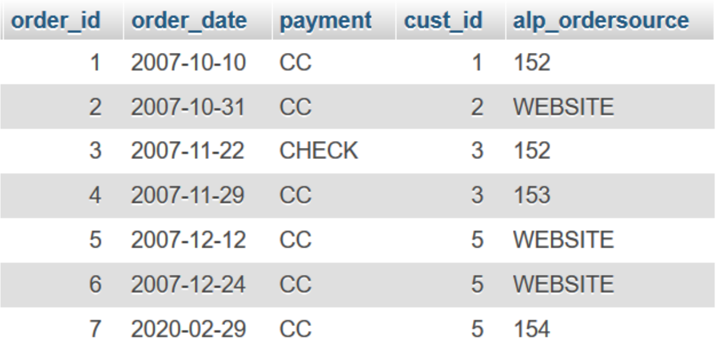
```
CALL sp_InsertOrder('2020-2-29','CC',5,'154')
```

Before:



order_id	order_date	payment	cust_id	alp_ordersource
1	2007-10-10	CC	1	152
2	2007-10-31	CC	2	WEBSITE
3	2007-11-22	CHECK	3	152
4	2007-11-29	CC	3	153
5	2007-12-12	CC	5	WEBSITE
6	2007-12-24	CC	5	WEBSITE

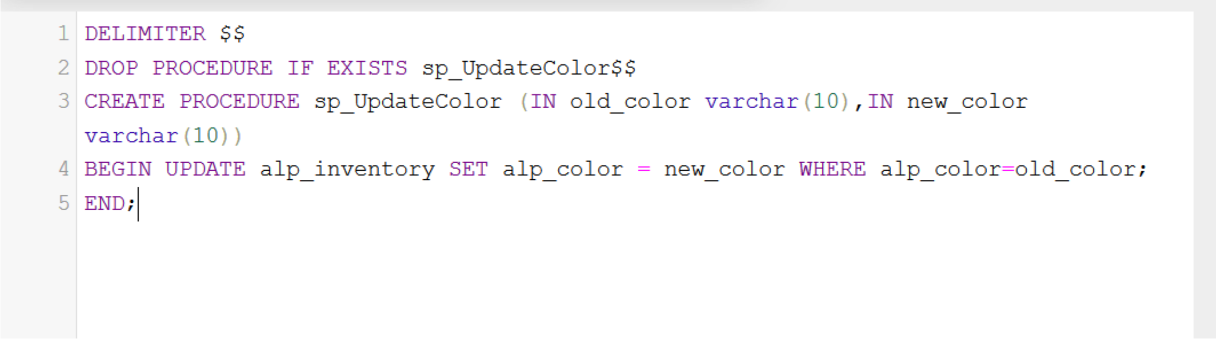
After:



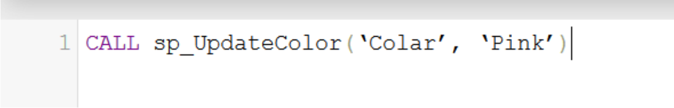
order_id	order_date	payment	cust_id	alp_ordersource
1	2007-10-10	CC	1	152
2	2007-10-31	CC	2	WEBSITE
3	2007-11-22	CHECK	3	152
4	2007-11-29	CC	3	153
5	2007-12-12	CC	5	WEBSITE
6	2007-12-24	CC	5	WEBSITE
7	2020-02-29	CC	5	154

Procedure 3:

Write a procedure that will allow a specified color in the inventory table to be changed to a different color (Hint: you will use the UPDATE command). This procedure will be passed two values: the old color and the new color. Test the procedure by changing the color ‘Coral’ to the color ‘Pink’. Name the procedure sp_UpdateColor

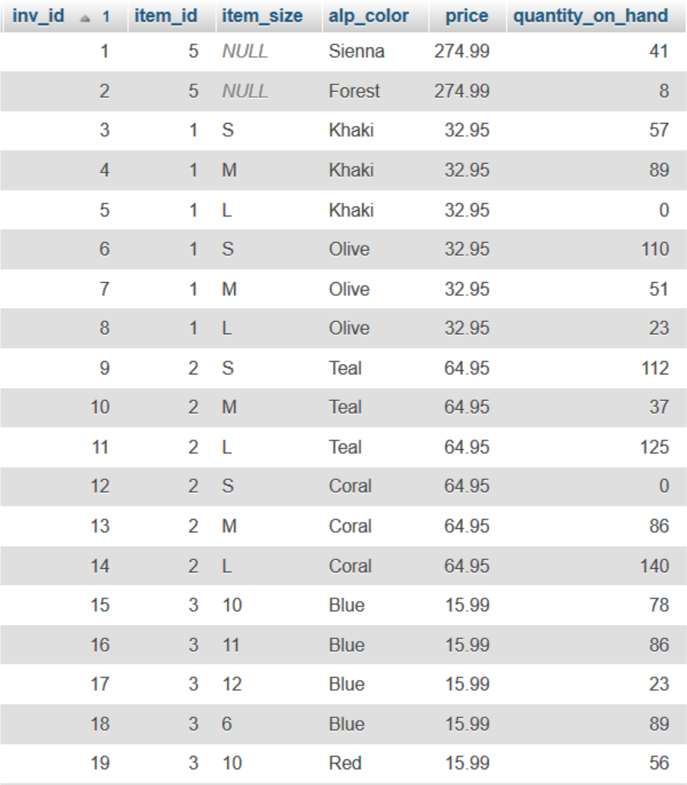


```
DELIMITER $$
DROP PROCEDURE IF EXISTS sp_UpdateColor$$
CREATE PROCEDURE sp_UpdateColor (IN old_color varchar(10),IN new_color varchar(10))
BEGIN UPDATE alp_inventory SET alp_color = new_color WHERE alp_color=old_color;
END;
```



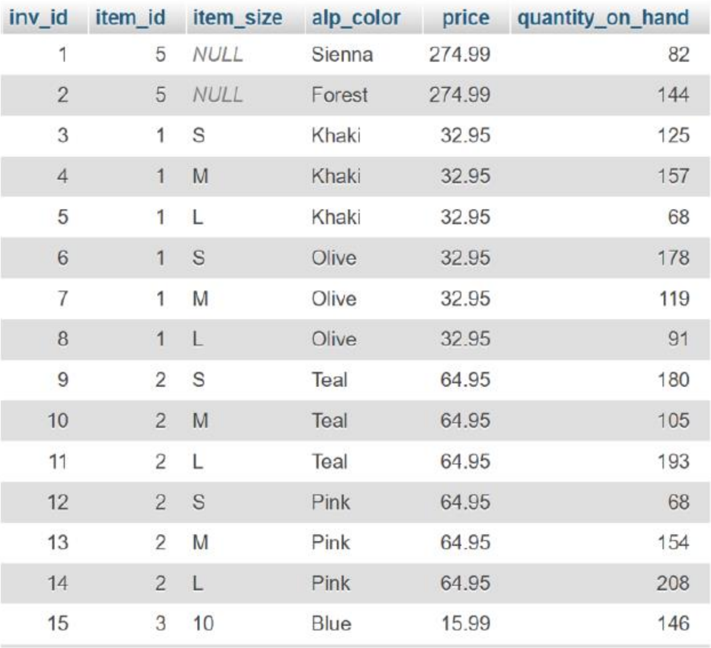
```
CALL sp_UpdateColor(‘Colar’, ‘Pink’)
```

Before:



inv_id ▲ 1	item_id	item_size	alp_color	price	quantity_on_hand
1	5	*NULL*	Sienna	274.99	41
2	5	*NULL*	Forest	274.99	8
3	1	S	Khaki	32.95	57
4	1	M	Khaki	32.95	89
5	1	L	Khaki	32.95	0
6	1	S	Olive	32.95	110
7	1	M	Olive	32.95	51
8	1	L	Olive	32.95	23
9	2	S	Teal	64.95	112
10	2	M	Teal	64.95	37
11	2	L	Teal	64.95	125
12	2	S	Coral	64.95	0
13	2	M	Coral	64.95	86
14	2	L	Coral	64.95	140
15	3	10	Blue	15.99	78
16	3	11	Blue	15.99	86
17	3	12	Blue	15.99	23
18	3	6	Blue	15.99	89
19	3	10	Red	15.99	56

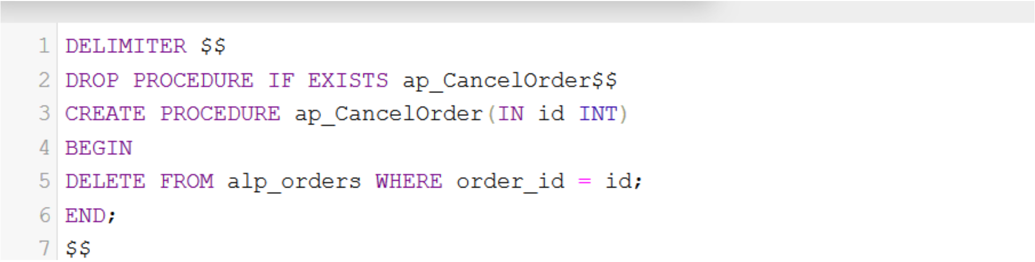
After:



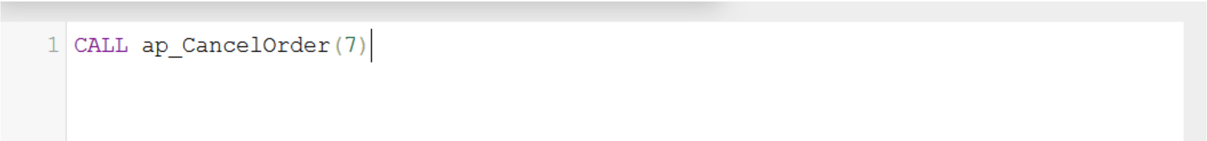
inv_id	item_id	item_size	alp_color	price	quantity_on_hand
1	5	NULL	Sienna	274.99	82
2	5	NULL	Forest	274.99	144
3	1	S	Khaki	32.95	125
4	1	M	Khaki	32.95	157
5	1	L	Khaki	32.95	68
6	1	S	Olive	32.95	178
7	1	M	Olive	32.95	119
8	1	L	Olive	32.95	91
9	2	S	Teal	64.95	180
10	2	M	Teal	64.95	105
11	2	L	Teal	64.95	193
12	2	S	Pink	64.95	68
13	2	M	Pink	64.95	154
14	2	L	Pink	64.95	208
15	3	10	Blue	15.99	146

Procedure 4

Write a procedure that will allow a user to cancel an order. The procedure will be passed an order_id and the necessary information will be deleted to cancel the order. Name the procedure sp_CancelOrder.

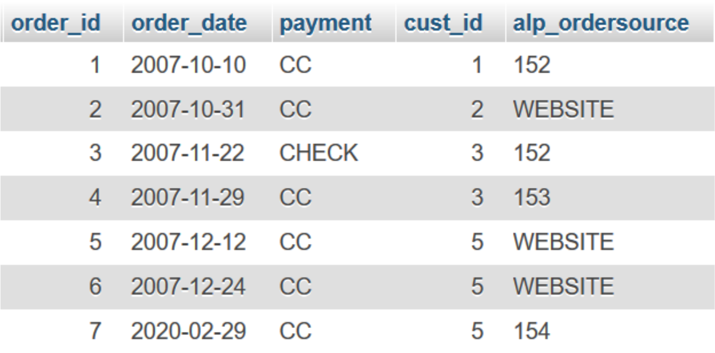


```
DELIMITER $$
DROP PROCEDURE IF EXISTS ap_CancelOrder$$
CREATE PROCEDURE ap_CancelOrder(IN id INT)
BEGIN
DELETE FROM alp_orders WHERE order_id = id;
END;
$$
```



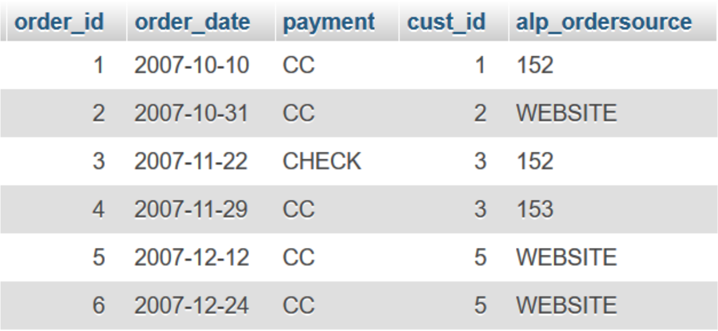
```
CALL ap_CancelOrder(7)
```

Before:



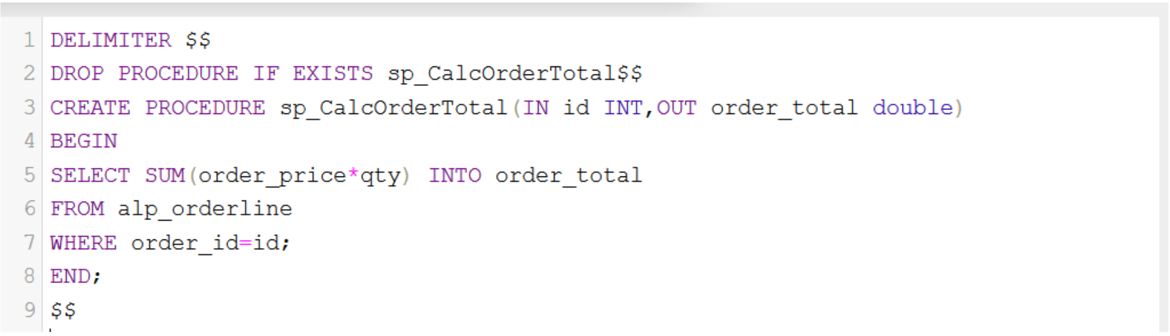
order_id	order_date	payment	cust_id	alp_ordersource
1	2007-10-10	CC	1	152
2	2007-10-31	CC	2	WEBSITE
3	2007-11-22	CHECK	3	152
4	2007-11-29	CC	3	153
5	2007-12-12	CC	5	WEBSITE
6	2007-12-24	CC	5	WEBSITE
7	2020-02-29	CC	5	154

After:

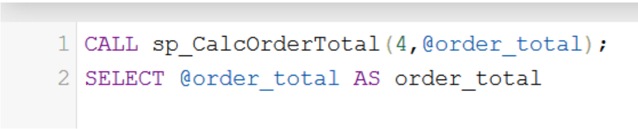


order_id	order_date	payment	cust_id	alp_ordersource
1	2007-10-10	CC	1	152
2	2007-10-31	CC	2	WEBSITE
3	2007-11-22	CHECK	3	152
4	2007-11-29	CC	3	153
5	2007-12-12	CC	5	WEBSITE
6	2007-12-24	CC	5	WEBSITE

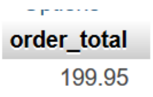
Procedure 5: Write a procedure that will calculate the total for a specified order_id. The procedure will receive one input parameter (order_id) and return one parameter (order_total). Name the procedure sp_CalcOrderTotal.



```
DELIMITER $$
DROP PROCEDURE IF EXISTS sp_CalcOrderTotal$$
CREATE PROCEDURE sp_CalcOrderTotal(IN id INT,OUT order_total double)
BEGIN
SELECT SUM(order_price*qty) INTO order_total
FROM alp_orderline
WHERE order_id=id;
END;
$$
```

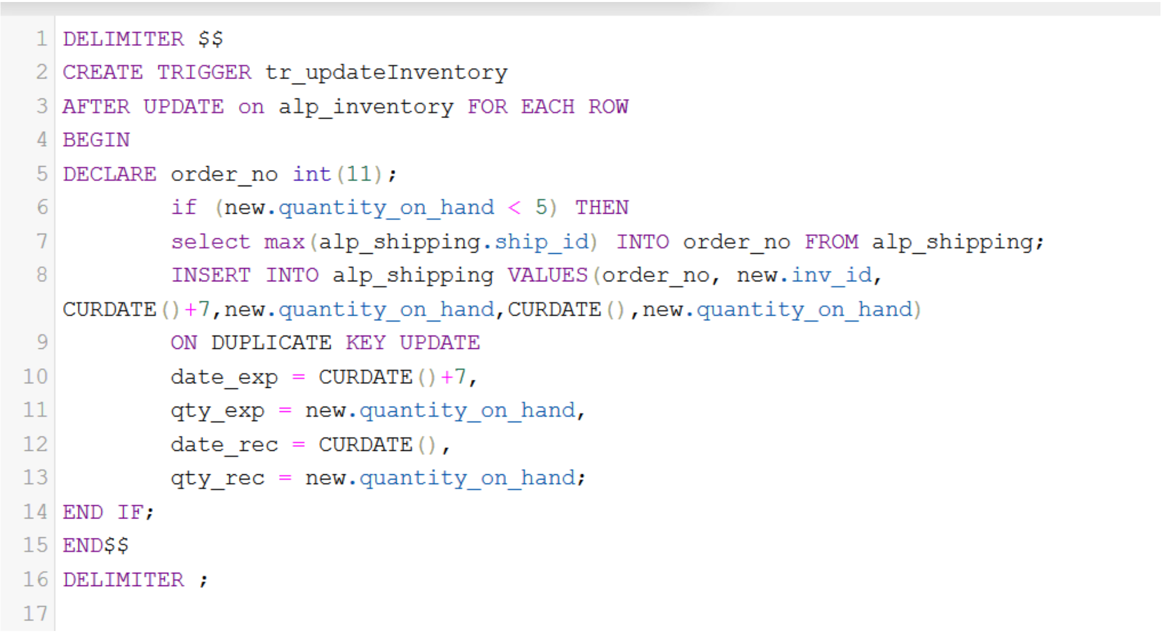


```
CALL sp_CalcOrderTotal(4,@order_total);
SELECT @order_total AS order_total
```

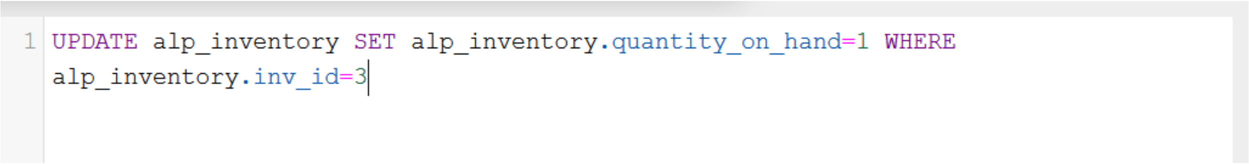


order_total
199.95

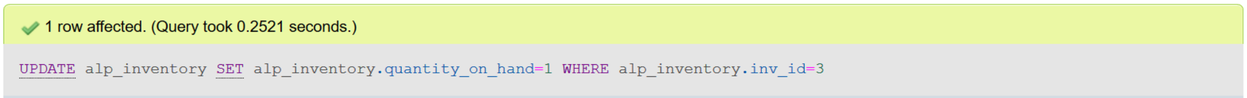
Trigger 1: Create a trigger that will automatically create a shipping record if an inventory item is update with a quantity_on_hand < 5. You can hard code the date_expected or research on the Internet how to get the current date from the system and add 7 days to it for the date expected. Name the trigger: tr_updateInventory.



```
DELIMITER $$
CREATE TRIGGER tr_updateInventory
AFTER UPDATE on alp_inventory FOR EACH ROW
BEGIN
DECLARE order_no int(11);
        if (new.quantity_on_hand < 5) THEN
        select max(alp_shipping.ship_id) INTO order_no FROM alp_shipping;
        INSERT INTO alp_shipping VALUES(order_no, new.inv_id,
CURDATE()+7,new.quantity_on_hand,CURDATE(),new.quantity_on_hand)
        ON DUPLICATE KEY UPDATE
        date_exp = CURDATE()+7,
        qty_exp = new.quantity_on_hand,
        date_rec = CURDATE(),
        qty_rec = new.quantity_on_hand;
END IF;
END$$
DELIMITER ;
```

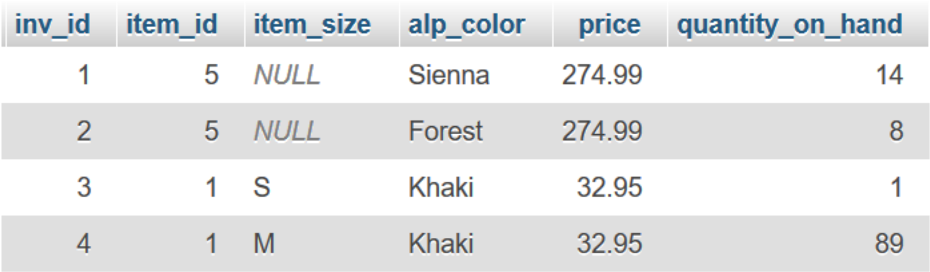


```
UPDATE alp_inventory SET alp_inventory.quantity_on_hand=1 WHERE
alp_inventory.inv_id=3
```

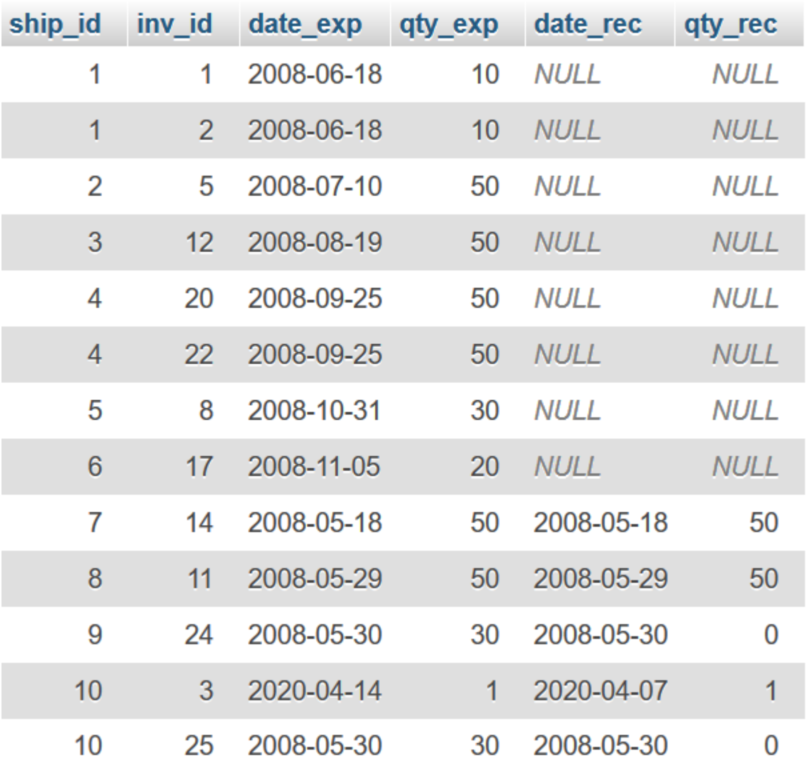


```
✓ 1 row affected. (Query took 0.2521 seconds.)

UPDATE alp_inventory SET alp_inventory.quantity_on_hand=1 WHERE alp_inventory.inv_id=3
```



inv_id	item_id	item_size	alp_color	price	quantity_on_hand
1	5	*NULL*	Sienna	274.99	14
2	5	*NULL*	Forest	274.99	8
3	1	S	Khaki	32.95	1
4	1	M	Khaki	32.95	89



ship_id	inv_id	date_exp	qty_exp	date_rec	qty_rec
1	1	2008-06-18	10	*NULL*	*NULL*
1	2	2008-06-18	10	*NULL*	*NULL*
2	5	2008-07-10	50	*NULL*	*NULL*
3	12	2008-08-19	50	*NULL*	*NULL*
4	20	2008-09-25	50	*NULL*	*NULL*
4	22	2008-09-25	50	*NULL*	*NULL*
5	8	2008-10-31	30	*NULL*	*NULL*
6	17	2008-11-05	20	*NULL*	*NULL*
7	14	2008-05-18	50	2008-05-18	50
8	11	2008-05-29	50	2008-05-29	50
9	24	2008-05-30	30	2008-05-30	0
10	3	2020-04-14	1	2020-04-07	1
10	25	2008-05-30	30	2008-05-30	0