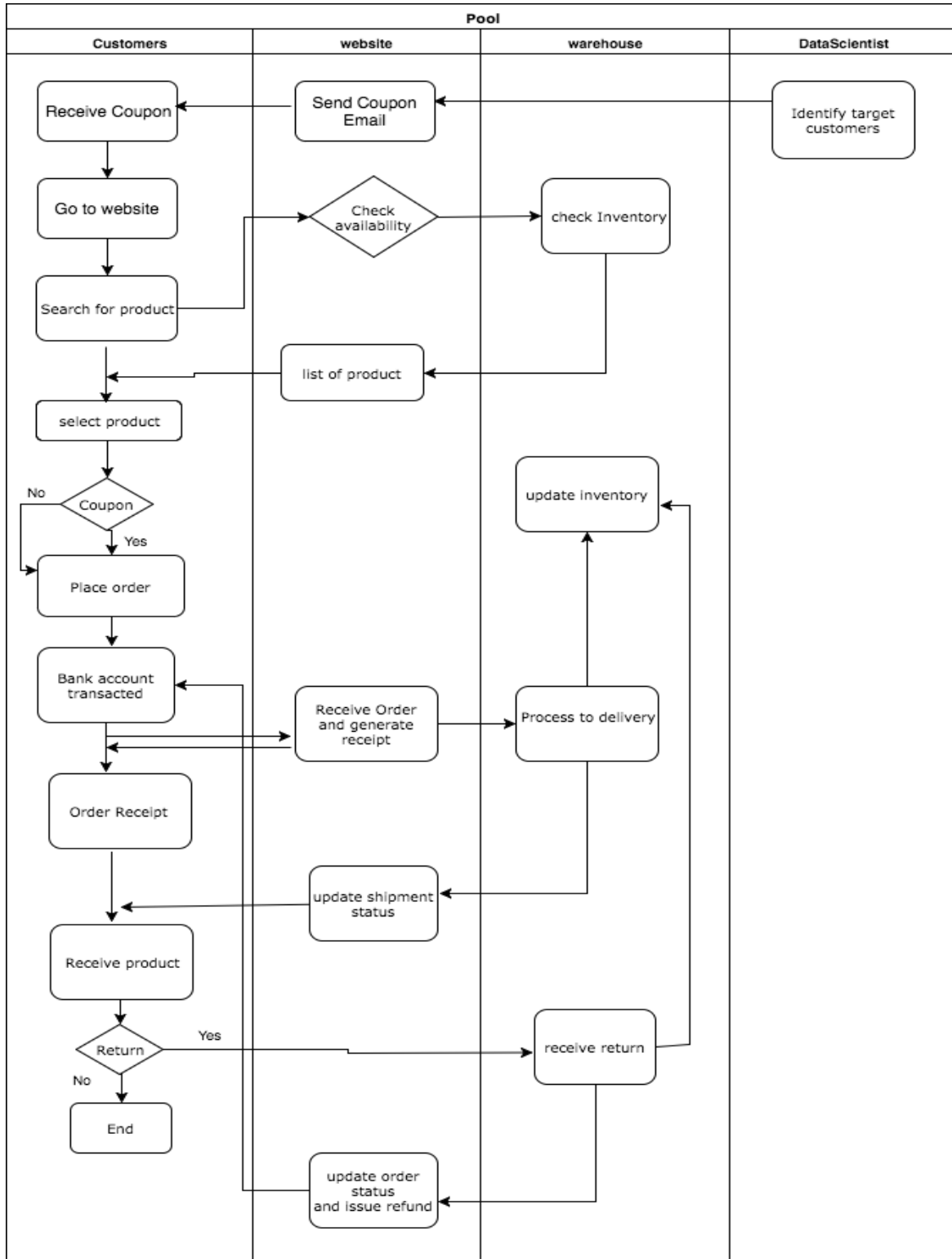
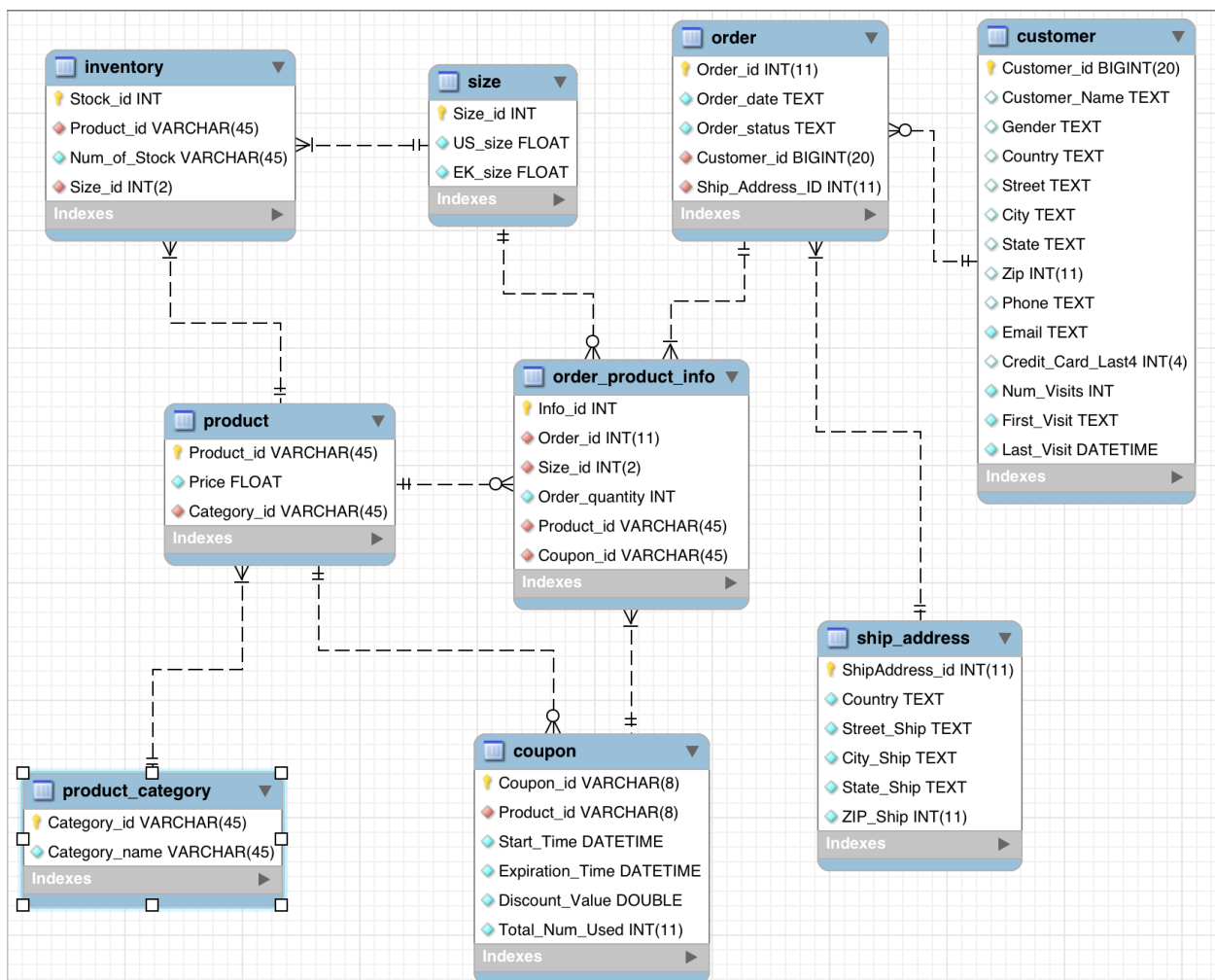


Swim Lane



Logical Schema – ERD



Physical Schema - Database Dictionary

Customer			
<i>Name</i>	<i>Type</i>	<i>Constraint</i>	<i>Description</i>
Customer_id	INT	PRIMARY KEY	
Customer_Name	VARCHAR(45)	NOT NULL	
Gender	VARCHAR(45)	NOT NULL	
Country	VARCHAR(12)	NOT NULL	
Street	VARCHAR(45)	NOT NULL	
City	VARCHAR(45)	NOT NULL	
State	VARCHAR(45)	NOT NULL	
ZIP	INT	NOT NULL	
Phone	VARCHAR(45)	NOT NULL	
Email	VARCHAR(45)	NOT NULL	
First_Visit	VARCHAR(45)	NOT NULL	
Last_Visit	Datetime	NOT NULL	
Num_of_Visit	INT	NOT NULL	
Credit_card_last4	INT	NOT NULL	Last four digits of credit card

Coupon			
<i>Name</i>	<i>Type</i>	<i>Constraint</i>	<i>Description</i>
Coupon_id	VARCHAR(8)	PRIMARY KEY	
Product_id	INT(11)	FOREIGN KEY, NOT NULL	
Start_Time	VARCHAR(45)	NOT NULL	

Expiration_Tlme	VARCHAR(45)	NOT NULL	
Discount_Value	FLOAT	NOT NULL	Percentage off as a decimal amount
Total_Num_Used	INT(11)	NOT NULL	Record the total number of coupon used before expiration time

Order			
<i>Name</i>	<i>Type</i>	<i>Constraint</i>	<i>Description</i>
Order_id	INT	PRIMARY KEY	This attribute will be indexed
Order_date	VARCHAR(45)	NOT NULL	
Order_status	VARCHAR(45)	NOT NULL	
Customer_id	INT	FOREIGN KEY, NOT NULL	
Ship_Address_id	INT	FOREIGN KEY, NOT NULL	

Order_Product_Info			
<i>Name</i>	<i>Type</i>	<i>Constraint</i>	<i>Description</i>
Info_id	INT(11)	PRIMARY KEY	
Order_id	INT(11)	FOREIGN KEY KEY	
Order_quantity	INT(11)	NOT NULL	
Product_id	INT(11)	FOREIGN KEY, NOT NULL	
Size_id	INT(11)	FOREIGN KEY, NOT NULL	

Coupon_id	VARCHAR(8)	FOREIGN KEY, NULL	
-----------	------------	-------------------	--

Ship_Address			
<i>Name</i>	<i>Type</i>	<i>Constraint</i>	<i>Description</i>
Ship_Address_ID	INT	PRIMARY KEY	
Country	VARCHAR(12)	NOT NULL	
Street	VARCHAR(45)	NOT NULL	
City	VARCHAR(45)	NOT NULL	
State	VARCHAR(45)	NOT NULL	
ZIP	INT	NOT NULL	

Product			
<i>Name</i>	<i>Type</i>	<i>Constraint</i>	<i>Description</i>
Product_id	INT(11)	PRIMARY KEY	This attribute will be indexed
Price	FLOAT	NOT NULL	
Category_id	VARCHAR(45)	FOREIGN KEY, NOT NULL	

Product_Description			We will use NoSQL for this table
<i>Name</i>	<i>Type</i>	<i>Constraint</i>	<i>Description</i>
Product_id	INT(11)	PRIMARY KEY	
Product_description	Text	NOT NULL	
Product_image	Photos	NOT NULL	

Product_Category			
<i>Name</i>	<i>Type</i>	<i>Constraint</i>	<i>Description</i>

Category_id	VARCHAR(45)	PRIMARY KEY	
Category_name	VARCHAR(45)	NOT NULL	Pumps, sandals, etc.

Review			We will use NoSQL for this table
<i>Name</i>	<i>Type</i>	<i>Constraint</i>	<i>Description</i>
Review_id	INT(11)	PRIMARY KEY	
Username	VARCHAR(45)	NULL	
Product_id	INT(11)	FOREIGN KEY, NOT NULL	
Customer_id	INT(11)	FOREIGN KEY, NOT NULL	
Date	Text	NOT NULL	
Rating	INT(11)	NULL	
Content	Text	NOT NULL	
Image	Photos	NULL	
Useful	INT(11)	NULL	
Funny	INT(11)	NULL	
Cool	INT(11)	NULL	

Inventory			
<i>Name</i>	<i>Type</i>	<i>Constraint</i>	<i>Description</i>
Stock_id	INT(11)	PRIMARY KEY	
Product_id	INT(11)	FOREIGN KEY, NOT NULL	
Size_id	INT(11)	FOREIGN KEY, NOT NULL	
Num_of_Stock	INT	NOT NULL	

Size			
-------------	--	--	--

<i>Name</i>	<i>Type</i>	<i>Constraint</i>	<i>Description</i>
Size_id	INT(11)	PRIMARY KEY	
US_size	FLOAT(11)	NOT NULL	
UK_size	FLOAT(11)	NOT NULL	

Queries

Customers

1. View list of products, product price and product picture.

First, select product and product price in MySQL

```
select p.Product_id, p.Price, pc.Category_name
```

```
from product p
```

```
left join product_category pc
```

```
on p.Category_id = pc.category_id;
```

89	•	select p.Product_id, p.Price, pc.Category_name
90		from product p
91		left join product_category pc on p.Category_id = pc.category_id;
92		
93		

100%	31:91	1 error found
Result Grid	Filter Rows: <input type="text" value="Search"/>	Export:
Product_id	Price	Category_name
1	80	Sandals
2	108	Sandals
3	92	Pumps
4	165	Pumps
5	95	Athletic shoes
6	64	Athletic shoes
7	55	Boots
8	123	Boots
9	71	Flats
10	88	Flats

At the same time, query product pic from NoSQL database:

```
>db.Product_Description.find({}, {Product_id:true,Product_image:true,_id:false}).pretty()  
pretty()
```

```
> db.Product_Description.find({}, {Product_id:true,Product_image:true,_id:false}).pretty()  
{  
  "Product_id" : 1,  
  "Product_image" : "/Users/alicewu/Documents/SCU/SQL/project/image/p1.png"  
}  
{  
  "Product_id" : 2,  
  "Product_image" : "/Users/alicewu/Documents/SCU/SQL/project/image/p2.png"  
}  
{  
  "Product_id" : 3,  
  "Product_image" : "/Users/alicewu/Documents/SCU/SQL/project/image/p3.png"  
}  
{  
  "Product_id" : 4,  
  "Product_image" : "/Users/alicewu/Documents/SCU/SQL/project/image/p4.png"  
}  
{  
  "Product_id" : 5,  
  "Product_image" : "/Users/alicewu/Documents/SCU/SQL/project/image/p5.png"  
}
```

2. To view the detail of one product .

First, query from MySQL:

```
select m.Product_id, m.US_size, m.Price ,m.Num_of_Stock, pc.Category_name  
from (select n.Product_id, n.US_size, p.Price ,n.Num_of_Stock,p.Category_id  
from (select i.Product_id, s.US_size, i.Num_of_Stock  
      from inventory i  
      left join size s  
      on i.Size_id = s.Size_id) n  
      left join product p  
      on n.Product_id = p.Product_id  
      where n.Product_id = 1) m  
left join product_category pc  
on m.Category_id = pc.Category_id;
```



```

106 • select m.Product_id, m.US_size, m.Price ,m.Num_of_Stock, pc.Category_name
107 from (select n.Product_id, n.US_size, p.Price ,n.Num_of_Stock,p.Category_id
108 from (select i.Product_id, s.US_size, i.Num_of_Stock
109 from inventory i
110 left join size s
111 on i.Size_id = s.Size_id) n
112 left join product p
113 on n.Product_id = p.Product_id
114 where n.Product_id = 1) m
115 left join product_category pc
116 on m.Category_id = pc.Category_id;
117
118

```

100% 30:115 1 error found

Result Grid Filter Rows: Search Export:

	Product_id	US_size	Price	Num_of_Stock	Category_name
▶	1	5	80	84	Sandals
	1	6	80	84	Sandals
	1	7	80	7	Sandals
	1	8	80	27	Sandals
	1	9	80	72	Sandals

At the same time, query for product details and product picture from NoSQL database:

```
> db.Product_Description.find({Product_id:{Seq:1}}).pretty()
```

```

> db.Product_Description.find({Product_id:{Seq:1}}).pretty()
{
  "_id" : ObjectId("5aa9f9a21025d6b2b1560534"),
  "Product_id" : 1,
  "Product_description" : "Synthetic\rSynthetic sole\rHeel measures approximately 3/4\"\rPlatform measures approximately 1/2\"\rThong sandal with adjustable buckle straps featuring contoured cork footbed and grippy outsole",
  "Product_image" : "/Users/alicewu/Documents/SCU/SQL/project/image/p1.png"
}

```

3. Search for products by size, price, and category - Size 6, under \$100, boots

```

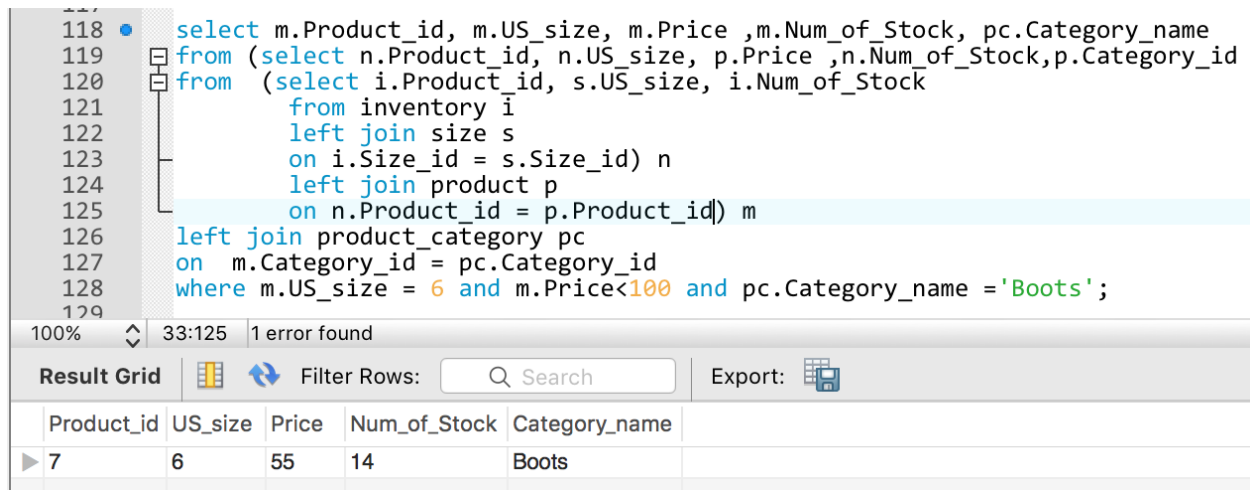
SELECT m.Product_id, m.US_size, m.Price ,m.Num_of_Stock, pc.Category_name
FROM (select n.Product_id, n.US_size, p.Price ,n.Num_of_Stock,p.Category_id
from (select i.Product_id, s.US_size, i.Num_of_Stock

```

```

from inventory i
left join size s
on i.Size_id = s.Size_id) n
left join product p
on n.Product_id = p.Product_id) m
LEFT JOIN product_category pc
ON m.Category_id = pc.Category_id
WHERE m.US_size = 6 and m.Price<100 and pc.Category_name ='Boots';

```



The screenshot shows a SQL IDE with a query editor on the left and a result grid on the right. The query is a complex JOIN statement. The result grid shows one row of data.

```

118 • select m.Product_id, m.US_size, m.Price ,m.Num_of_Stock, pc.Category_name
119 from (select n.Product_id, n.US_size, p.Price ,n.Num_of_Stock,p.Category_id
120 from (select i.Product_id, s.US_size, i.Num_of_Stock
121 from inventory i
122 left join size s
123 on i.Size_id = s.Size_id) n
124 left join product p
125 on n.Product_id = p.Product_id) m
126 left join product_category pc
127 on m.Category_id = pc.Category_id
128 where m.US_size = 6 and m.Price<100 and pc.Category_name ='Boots';
129

```

100% 33:125 1 error found

Result Grid Filter Rows: Search Export:

Product_id	US_size	Price	Num_of_Stock	Category_name
7	6	55	14	Boots

4. See product reviews. (from NoSQL)

```
> db.customers_reviews.find({product_id:{$eq:7}}).pretty()
```

```
> db.customers_reviews.find({product_id:{seq:7}}).pretty()
{
  "_id" : ObjectId("5aa801e43e888edcb8a388e1"),
  "review_id" : "A5500067",
  "customer_name" : "Wanda Goodwin",
  "product_id" : 7,
  "rating" : 4,
  "date" : "11/29/17 4:01",
  "text" : "Love these sandals- this color is sooo bright, a little too bright but probably why they wer
e such a great deal.\r",
  "useful" : 2,
  "funny" : 1,
  "cool" : 1
}
```

5. Update customer information - Customer Justine Gates wants to change her maiden name (Gates) to married name Cooper

Before UPDATE:

Select * FROM customer order by customer_name;

81560736599	Geraldine Buch...	Female	United States	P.O. Box 650, 2538 Mi. Av.
10769974799	Jackson Estrada	Male	United States	6512 Eu Av.
19417471899	Jerry Huff	Male	United States	1205 Scelerisque Street
87972646599	Jocelyn Hayden	Female	United States	8019 Eu Rd.
44947825699	Jordan Mooney	Female	United States	8423 Magna. Ave
67471462799	Judith Sexton	Female	United States	P.O. Box 630, 1222 Aliquet Ave
23547986539	Julia Wang	Female	United States	2500 Broadway
70897853699	Justine Gates	Female	United States	Ap #647-2055 Malesuada Av.
27373156899	Kylie Gomez	Female	United States	P.O. Box 409, 8542 Odio Street

UPDATE customer SET customer_name = "Justine Cooper" WHERE customer_name = "Justine Gates";

After UPDATE:

Select * FROM customer order by customer_name;

81560736599	Geraldine Buch...	Female	United States	P.O. Box 650, 2538 Mi. Av.	Newark
10769974799	Jackson Estrada	Male	United States	6512 Eu Av.	San Jose
19417471899	Jerry Huff	Male	United States	1205 Scelerisque Street	Newark
87972646599	Jocelyn Hayden	Female	United States	8019 Eu Rd.	Santa Clara
44947825699	Jordan Mooney	Female	United States	8423 Magna. Ave	San Diego
67471462799	Judith Sexton	Female	United States	P.O. Box 630, 1222 Aliquet Ave	Houston
23547986539	Julia Wang	Female	United States	2500 Broadway	Lubbock
70897853699	Justine Cooper	Female	United States	Ap #647-2055 Malesuada Av.	Seattle
27373156899	Kylie Gomez	Female	United States	P.O. Box 409, 8542 Odio Street	Floyd
90571742199	Leandra McGow...	Female	United States	Ap #574-8364 Vestibulum Av.	Boise

6. Check order history and order status - Customer Bailey Rosa wants to check the statuses and shipping information of all her orders

```
SELECT customer_name, order_id, order_date, order_status, s.country,  
street_ship AS street, city_ship AS city, state_ship AS state, zip_ship AS zip  
FROM customer c JOIN `order` o  
ON o.customer_id = c.customer_id  
JOIN ship_address s ON s.shipaddress_id = o.ship_address_id  
WHERE customer_name = "Bailey Rosa";
```

Customer_Name	order_id	order_date	order_status	country	street	city	state	zip
Bailey Rosa	212	2017-12-20 1:34:07	delivered	United States	Ap #730-9809 Nec, St.	Santa Clara	California	95053
Bailey Rosa	236	2017-12-11 21:13:40	delivered	United States	Ap #730-9809 Nec, St.	Santa Clara	California	95053
Bailey Rosa	245	2017-04-10 5:31:36	delivered	United States	Ap #730-9809 Nec, St.	Santa Clara	California	95053

7. Place orders - Customer Suki Poole wants to place a new order on 3/1/18 10:38 for quantity 1 of product 1 in a size 5, shipped her usual shipping address, with ship_id 117

```
INSERT INTO `order` (order_id, order_date, order_status, customer_id,  
ship_address_id)  
VALUES (250, "2018-03-01 10:38:00", "placed", 65941121199, 117);
```

```
INSERT INTO order_product_info  
VALUES (57, 250, 1, 1, 1, "");
```

Order table after insert:

Order_id	Order_date	Order_status	Customer_id	Ship_Address_ID
243	2017-06-30 1:50:35	delivered	56369599599	110
244	2017-12-31 2:22:09	shipped	90571742499	111
245	2017-04-10 5:31:36	delivered	37282242699	112
246	2017-11-24 3:57:56	delivered	87972646599	113
247	2017-03-31 21:01:02	delivered	67523638099	114
248	2017-12-29 13:33:57	placed	56236166699	115
249	2017-09-16 15:50:44	delivered	19417471899	104
250	2018-03-01 10:38:00	placed	65941121199	117

Order Product Information table after insert:

Info_id	Order_id	Order_quantity	Product_id	Size_id	Coupon_id
50	243	1	7	1	NULL
51	244	2	5	2	B8
52	245	2	5	3	B4
53	246	2	1	4	A6
54	247	1	2	5	NULL
55	248	1	6	1	A8
56	249	1	8	2	B6
57	250	1	1	1	

If Suki checks her order history after placing the order:

```
SELECT Customer_Name, order_id, order_date, order_status, s.country,
street_ship AS street, city_ship AS city, state_ship AS state, zip_ship AS zip
FROM customer c JOIN `order` o
ON o.customer_id = c.customer_id
JOIN ship_address s ON s.shipaddress_id = o.ship_address_id
WHERE customer_name = "Suki Poole";
```

Customer_Name	order_id	order_date	order_status	country	street	city	state	zip
Suki Poole	217	2017-10-21 6:35:07	delivered	United States	Ap #861-8992 Ligula Avenue	New York City	New York	10001
Suki Poole	250	2018-03-01 10:38:00	placed	United States	Ap #861-8992 Ligula Avenue	New York City	New York	10001

Warehouse

1. Check which orders have been placed but not yet been shipped

```
SELECT o.order_id, order_date, order_quantity, product_id, US_size, country,  
street_ship AS Street, city_ship AS City, state_ship AS State, zip_ship AS Zip  
FROM `order` o JOIN order_product_info pi  
ON o.order_id = pi.order_id  
LEFT JOIN size  
ON size.Size_id = pi.Size_id  
JOIN ship_address s  
ON s.shipaddress_id = o.ship_address_id  
WHERE order_status='placed';
```

order_id	order_date	order_quantity	product_id	US_size	country	Street	City	State	Zip
248	2017-12-29 13:33:57	1	6	5	United States	Ap #716-8128 Tempus, Road	Boise	Idaho	2991
250	2018-03-01 10:38:00	1	1	5	United States	Ap #861-8992 Ligula Avenue	New York City	New York	10001
211	2017-12-28 4:57:14	1	4	9	United States	Ap #574-8364 Vestibulum, Av.	Boise	Idaho	2991

2. Check number of stock for each product and size

```
SELECT Product_id, US_size, UK_size, Num_of_stock AS 'Quantity in Stock'  
FROM inventory i  
LEFT JOIN size  
ON i.Size_id = size.Size_id;
```

Product_id	US_size	UK_size	Quantity in Stock				
1	5	3	84	6	7	5	52
2	5	3	24	7	7	5	50
3	5	3	61	8	7	5	47
4	5	3	80	9	7	5	20
5	5	3	72	10	7	5	26
6	5	3	98	1	8	6	27
7	5	3	54	2	8	6	40
8	5	3	50	3	8	6	98
9	5	3	41	4	8	6	21
10	5	3	46	5	8	6	19
1	6	4	84	6	8	6	26
2	6	4	17	7	8	6	69
3	6	4	76	8	8	6	37
4	6	4	28	9	8	6	15
5	6	4	52	10	8	6	69
6	6	4	51	1	9	7	72
7	6	4	14	2	9	7	99
8	6	4	86	3	9	7	86
9	6	4	79	4	9	7	65
10	6	4	76	5	9	7	83
1	7	5	7	6	9	7	30
2	7	5	2	7	9	7	34
3	7	5	90	8	9	7	81
4	7	5	83	9	9	7	70
5	7	5	84	10	9	7	4

3. Delete cancelled orders

Before the delete:

```
SELECT * FROM `order` WHERE order_status = 'cancelled';
```

Order_id	Order_date	Order_status	Customer_id	Ship_Address_ID
203	2017-06-18 9:28:50	cancelled	81560736599	103
227	2017-09-27 4:56:55	cancelled	34827140399	106

```
DELETE FROM `order` WHERE order_status = 'cancelled';
```

374 18:58:46 DELETE FROM `order` WHERE order_status = 'cancelled' 2 row(s) affected 0.063 sec

After the delete:

```
SELECT count(order_status) FROM `order` WHERE order_status = 'cancelled';
```

count(order_status)
0

Marketing Analyst

1. Identify which coupons had the most uses

```
SELECT * FROM coupon
```

```
WHERE total_num_used = (SELECT MAX(total_num_used) FROM coupon);
```

Coupon_ID	Product_ID	Start_Time	Expiration_Time	Discount_Value	Total_Num_Used
B6	8	9/1/17	09/31/17	0.1	4
B8	5	12/1/17	12/31/17	0.2	4
A8	6	12/1/17	12/31/17	0.3	4

2. Find customers who are promotion sensitive (only buy with coupons)

```
SELECT nn.customer_id, nn.oc AS 'Number of Coupons Used' FROM ((select  
customer_id, COUNT(distinct o.order_id) AS oc
```

```
FROM `order` o join order_product_info opi on o.order_id = opi.order_id
```

```
WHERE coupon_id IS NOT NULL
```

```
GROUP BY customer_id) nn JOIN
```

```
(SELECT customer_id, COUNT(distinct order_id) AS ao FROM `order` GROUP BY  
customer_id) a ON nn.customer_id = a.customer_id)
```

```
WHERE nn.oc = a.ao;
```

customer_id	Number of Coupons Used
10769974799	1
21323718899	1
27373156899	1
37282242699	3
38864270899	1
39626886999	1
87972646599	3

3. Identify customers who have visited the website but have not placed an order


```
SELECT c.Customer_id, Customer_name FROM `order` o RIGHT JOIN customer c
ON o.customer_id = c.customer_id
WHERE order_id IS NULL;
```

Customer_id	Customer_name
81560736599	Geraldine Buchanan
41524398066	Rebaca Tom
23547986539	Julia Wang
32185743053	Alice Jeff

4. Identify dormant customers

```
SELECT a.customer_id, a.date_m
FROM(
SELECT customer_id, COUNT(customer_id) AS size, MAX(MONTH(Order_date)) AS
date_m
FROM `order`
GROUP BY customer_id ) a
WHERE 12 - date_m >= 6;
```

customer_id	date_m
17686436099	6
31642537899	2
34827140399	4
37875593499	5
39626886999	4
56369599599	6
67523638099	3

5. Check inventory to see which products have too much in stock (more than 300), and therefore could benefit from a coupon marketing campaign

```
SELECT Product_id, SUM(Num_of_stock) AS `Quantity in Stock` FROM inventory
GROUP BY Product_id
```

HAVING SUM(Num_of_stock) >300;

Product_id	Quantity in Stock
3	411
5	310
8	301

Indexes

CREATE INDEX order_index ON `order` (order_id);

SHOW INDEX FROM `order`;

	Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment
	order	1	order index	1	Order id	A	50	NULL	NULL	YES	BTREE		

CREATE INDEX product_index ON product (product_id);

SHOW INDEX FROM product;

	Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment
	product	1	product index	1	Product id	A	10	NULL	NULL	YES	BTREE		

Views

Customer : Order confirmation View

CREATE VIEW Customer_view AS

```
select o.Order_id, p.Product_id, p.Price, op.Order_quantity, c.Coupon_ID,
c.Discount_Value, IF(c.Coupon_ID is not null, p.Price*op.Order_quantity*(1-
c.Discount_Value),p.Price*op.Order_quantity) as payment, o.Order_status
```

```
from `order` o
```

```
LEFT JOIN order_product_info op
```

```
on o.Order_id = op.Order_id
```

```
LEFT JOIN product p
```

```
on op.Product_id = p.Product_id
```

```
LEFT JOIN coupon c
```

on op.Coupon_id = c.Coupon_ID

group by o.Order_id;

select * from Customer_view;

Order_id	Product_id	Price	Order_quantity	Coupon_ID	Discount_Value	payment	Order_status
200	3	92	1	NULL	NULL	92	delivered
201	5	95	1	B8	0.2	76	shipped
202	2	108	2	B7	0.2	172.8	delivered
204	2	108	1	A3	0.05	102.6	delivered
205	8	123	1	B9	0.2	98.4	delivered
206	4	165	1	NULL	NULL	165	delivered
207	3	92	1	NULL	NULL	92	delivered
208	9	71	2	NULL	NULL	142	delivered
209	8	123	3	B9	0.2	295.2	shipped
210	1	10	1	A1	0.05	76	delivered
211	4	165	1	NULL	NULL	165	placed
212	5	95	1	B8	0.2	76	delivered
213	6	64	1	A8	0.3	44.8	delivered
214	7	55	2	NULL	NULL	110	delivered
215	3	92	2	NULL	NULL	184	delivered
216	8	123	1	B6	0.1	110.7	delivered
217	2	108	1	NULL	NULL	108	delivered
218	1	80	1	NULL	NULL	80	delivered

Warehouse view:

The warehouse needs to be able to see all the orders, what products have been ordered and their current inventory, and who and where to ship the orders:

CREATE VIEW Warehouse_view AS

SELECT o.Order_id, o.Order_date, o.Order_status, o.Customer_id,
Customer_name, i.Product_id, i.Size_id, Num_of_Stock, ShipAddress_id,
s.Country, Street_Ship, City_ship, State_ship, ZIP_ship

FROM `order` o JOIN customer c ON o.customer_id = c.customer_id

JOIN order_product_info op ON o.order_id = op.order_id

JOIN inventory i ON op.product_id = i.product_id

JOIN ship_address s ON s.ShipAddress_id = o.Ship_Address_ID;

SELECT * FROM Warehouse_view;

Order_id	Order_date	Order_status	Customer_id	Customer_name	Product_id	Size_id	Num_of_Stock	ShipAddress	Country	Street_Ship	City_ship	State_ship	ZIP_ship
202	11/18/2017 9:55	delivered	3.89E+10	Roanna Ayers	2	1	24	102 United Stat	363-5379	USeward	Alaska	99664	
202	11/18/2017 9:55	delivered	3.89E+10	Roanna Ayers	2	2	17	102 United Stat	363-5379	USeward	Alaska	99664	
202	11/18/2017 9:55	delivered	3.89E+10	Roanna Ayers	2	3	2	102 United Stat	363-5379	USeward	Alaska	99664	
202	11/18/2017 9:55	delivered	3.89E+10	Roanna Ayers	2	4	40	102 United Stat	363-5379	USeward	Alaska	99664	
202	11/18/2017 9:55	delivered	3.89E+10	Roanna Ayers	2	5	99	102 United Stat	363-5379	USeward	Alaska	99664	
204	4/18/2017 20:34	delivered	1.94E+10	Jerry Huff	2	1	24	104 United Stat	1205 Scele	Newark	Delaware	19711	
204	4/18/2017 20:34	delivered	1.94E+10	Jerry Huff	2	2	17	104 United Stat	1205 Scele	Newark	Delaware	19711	
204	4/18/2017 20:34	delivered	1.94E+10	Jerry Huff	2	3	2	104 United Stat	1205 Scele	Newark	Delaware	19711	
204	4/18/2017 20:34	delivered	1.94E+10	Jerry Huff	2	4	40	104 United Stat	1205 Scele	Newark	Delaware	19711	
204	4/18/2017 20:34	delivered	1.94E+10	Jerry Huff	2	5	99	104 United Stat	1205 Scele	Newark	Delaware	19711	
206	4/7/2017 0:55	delivered	3.48E+10	Wanda Goodwin	4	1	80	106 United Stat	P.O. Box 2C	Santa Clara	California	95053	
206	4/7/2017 0:55	delivered	3.48E+10	Wanda Goodwin	4	2	28	106 United Stat	P.O. Box 2C	Santa Clara	California	95053	
206	4/7/2017 0:55	delivered	3.48E+10	Wanda Goodwin	4	3	83	106 United Stat	P.O. Box 2C	Santa Clara	California	95053	
206	4/7/2017 0:55	delivered	3.48E+10	Wanda Goodwin	4	4	21	106 United Stat	P.O. Box 2C	Santa Clara	California	95053	
206	4/7/2017 0:55	delivered	3.48E+10	Wanda Goodwin	4	5	65	106 United Stat	P.O. Box 2C	Santa Clara	California	95053	
208	9/22/2017 10:54	delivered	7.09E+10	Justine Gates	9	1	41	108 United Stat	Ap #647-2C	Seattle	Washington	98105	
208	9/22/2017 10:54	delivered	7.09E+10	Justine Gates	9	2	79	108 United Stat	Ap #647-2C	Seattle	Washington	98105	
208	9/22/2017 10:54	delivered	7.09E+10	Justine Gates	9	3	20	108 United Stat	Ap #647-2C	Seattle	Washington	98105	
208	9/22/2017 10:54	delivered	7.09E+10	Justine Gates	9	4	15	108 United Stat	Ap #647-2C	Seattle	Washington	98105	
208	9/22/2017 10:54	delivered	7.09E+10	Justine Gates	9	5	70	108 United Stat	Ap #647-2C	Seattle	Washington	98105	
208	9/22/2017 10:54	delivered	7.09E+10	Justine Gates	6	1	98	108 United Stat	Ap #647-2C	Seattle	Washington	98105	
208	9/22/2017 10:54	delivered	7.09E+10	Justine Gates	6	2	51	108 United Stat	Ap #647-2C	Seattle	Washington	98105	
208	9/22/2017 10:54	delivered	7.09E+10	Justine Gates	6	3	52	108 United Stat	Ap #647-2C	Seattle	Washington	98105	
208	9/22/2017 10:54	delivered	7.09E+10	Justine Gates	6	4	26	108 United Stat	Ap #647-2C	Seattle	Washington	98105	
208	9/22/2017 10:54	delivered	7.09E+10	Justine Gates	6	5	30	108 United Stat	Ap #647-2C	Seattle	Washington	98105	
212	12/20/2017 1:34	delivered	3.73E+10	Bailey Rosa	5	1	72	112 United Stat	Ap #730-9E	Santa Clara	California	95053	
212	12/20/2017 1:34	delivered	3.73E+10	Bailey Rosa	5	2	52	112 United Stat	Ap #730-9E	Santa Clara	California	95053	
212	12/20/2017 1:34	delivered	3.73E+10	Bailey Rosa	5	3	84	112 United Stat	Ap #730-9E	Santa Clara	California	95053	
212	12/20/2017 1:34	delivered	3.73E+10	Bailey Rosa	5	4	19	112 United Stat	Ap #730-9E	Santa Clara	California	95053	
212	12/20/2017 1:34	delivered	3.73E+10	Bailey Rosa	5	5	83	112 United Stat	Ap #730-9E	Santa Clara	California	95053	
213	12/18/2017 19:13	delivered	8.8E+10	Jocelyn Hayden	6	1	98	113 United Stat	8019 Eu Rd	Santa Clara	California	95051	
213	12/18/2017 19:13	delivered	8.8E+10	Jocelyn Hayden	6	2	51	113 United Stat	8019 Eu Rd	Santa Clara	California	95051	
213	12/18/2017 19:13	delivered	8.8E+10	Jocelyn Hayden	6	3	52	113 United Stat	8019 Eu Rd	Santa Clara	California	95051	
213	12/18/2017 19:13	delivered	8.8E+10	Jocelyn Hayden	6	4	26	113 United Stat	8019 Eu Rd	Santa Clara	California	95051	

Marketing Analyst views:

1. Coupon Information View:

CREATE VIEW Coupon_view AS

select c.Coupon_ID, o.order_id, c.Discount_Value, p.Product_id, p.Price,
op.Order_quantity, p.Price*op.Order_quantity*(1-c.Discount_Value) as payment
from `order` o

LEFT JOIN order_product_info op

on o.order_id = op.order_id

LEFT JOIN product p

on op.product_id = p.product_id

LEFT JOIN coupon c

on op.coupon_id = c.coupon_id

where c.coupon_id is not null;

select * from Coupon_view;

Coupon_ID	Order_id	Discount_Value	Product_id	Price	Order_quantity	payment
A1	210	0.05	1	80	1	76
A2	219	0.15	3	92	1	78.2
A3	204	0.05	2	108	1	102.6
A3	237	0.05	4	165	1	156.75
B4	223	0.2	5	95	2	152
B4	240	0.2	5	95	1	76
B4	245	0.2	5	95	2	152
A5	235	0.15	8	123	1	104.55
B6	216	0.1	8	123	1	110.7
B6	222	0.1	8	123	1	110.7
B6	227	0.1	8	123	2	221.4
B6	249	0.1	8	123	1	110.7
A6	246	0.15	1	80	2	136
B7	202	0.2	2	108	2	172.8
A7	232	0.3	3	92	2	128.79...
B8	201	0.2	5	95	1	76
B8	212	0.2	5	95	1	76
B8	230	0.2	5	95	2	152
B8	244	0.2	5	95	2	152
A8	213	0.3	6	64	1	44.8
A8	224	0.3	6	64	4	179.2
A8	236	0.3	6	64	1	44.8
A8	248	0.3	6	64	1	44.8
B9	205	0.2	8	123	1	98.4
B9	209	0.2	8	123	3	295.2

2. Customer Order Information View:

CREATE VIEW Customer_order_info_view AS

select c.Customer_ID, c.Customer_Name, c.Gender, c.Country, c.Street, c.City,
c.State, c.ZIP, c.Email, c.Num_Visits,

c.First_Visit, c.Last_Visit, o.Order_id, p.Product_id, p.Price, op.Order_quantity,
co.Coupon_ID, co.Discount_Value,

IF(co.Coupon_ID is not null, p.Price*op.Order_quantity*(1-
co.Discount_Value),p.Price*op.Order_quantity) as payment, o.Order_status

from customer c

LEFT JOIN `order` o

on c.Customer_ID = o.Customer_ID

LEFT JOIN order_product_info op

on o.Order_id = op.Order_id

LEFT JOIN product p

on op.Product_id = p.Product_id

LEFT JOIN coupon co on op.Coupon_id = co.Coupon_ID;

select * from Customer_order_info_view;

Customer_ID	Customer_Name	Gender	Country	Street	City	State	ZIP	Email	Num_Visits	First_Visit
19417471899	Jerrv Huff	Male	United States	1205 Scelerisoue Street	Newark	Delaware	19711	dis@Sednulla.edu	7	1/1/17 7:22
87972646599	Jocelvyn Havden	Female	United States	8019 Eu Rd.	Santa Clara	California	95051	maona.a.tortor@sitametluctus.ora	16	5/20/16 17:24
37282242699	Bailev Rosa	Female	United States	Ao #730-9809 Nec. St.	Santa Clara	California	95053	Curabitur.dictum.Phasellus@ac.ca	2	9/19/16 19:51
78315977299	Charitv Landlev	Female	United States	109-7307 Cras Road	Houston	Texas	77001	Donec@Nuncmauris.com	8	6/10/16 16:16
19417471899	Jerrv Huff	Male	United States	1205 Scelerisoue Street	Newark	Delaware	19711	dis@Sednulla.edu	7	1/1/17 7:22
87972646599	Jocelvyn Havden	Female	United States	8019 Eu Rd.	Santa Clara	California	95051	maona.a.tortor@sitametluctus.ora	16	5/20/16 17:24
38864270899	Roanna Avers	Female	United States	363-5379 Ullamcorper. St.	Seward	Alaska	99664	Nam@vitae.co.uk	6	9/29/16 9:02
37282242699	Bailev Rosa	Female	United States	Ao #730-9809 Nec. St.	Santa Clara	California	95053	Curabitur.dictum.Phasellus@ac.ca	2	9/19/16 19:51
79797041399	Pamela Gilmore	Female	United States	Ao #532-9555 Fames Rd.	Jackson	Mississiooi	39056	maona@nondaobusrutrum.edu	20	3/11/16 11:11
87972646599	Jocelvyn Havden	Female	United States	8019 Eu Rd.	Santa Clara	California	95051	maona.a.tortor@sitametluctus.ora	16	5/20/16 17:24
10769974799	Jackson Estrada	Male	United States	6512 Eu Av.	San Jose	California	94088	imperdiet.dictum@poorttortellus...	6	8/26/16 14:15
37282242699	Bailev Rosa	Female	United States	Ao #730-9809 Nec. St.	Santa Clara	California	95053	Curabitur.dictum.Phasellus@ac.ca	2	9/19/16 19:51
34827140399	Wanda Goodwin	Female	United States	P.O. Box 203. 7026 Nisi ...	Santa Clara	California	95053	socis@orciacusvestibulum.co.uk	9	12/26/16 4:46
70897853699	Justine Gates	Female	United States	Ao #647-2055 Malesuad...	Seattle	Washington	98105	at.libero@luctusetultrices.edu	6	12/10/16 12:34
70897853699	Justine Gates	Female	United States	Ao #647-2055 Malesuad...	Seattle	Washington	98105	at.libero@luctusetultrices.edu	6	12/10/16 12:34
65941121199	Suki Poole	Female	United States	Ao #861-8992 Lioula Ave...	New York ...	New York	10001	nonummy@Proinvelnisl.ora	12	12/13/16 13:37
79797041399	Pamela Gilmore	Female	United States	Ao #532-9555 Fames Rd.	Jackson	Mississiooi	39056	maona@nondaobusrutrum.edu	20	3/11/16 11:11
17686436099	Malik Delacruz	Male	United States	580-7613 Nunc Road	Tallahassee	Florida	32301	dolor.Fusce@Mauris.co.uk	8	2/20/16 9:24
19417471899	Jerrv Huff	Male	United States	1205 Scelerisoue Street	Newark	Delaware	19711	dis@Sednulla.edu	7	1/1/17 7:22
70897853699	Justine Gates	Female	United States	Ao #647-2055 Malesuad...	Seattle	Washington	98105	at.libero@luctusetultrices.edu	6	12/10/16 12:34
17686436099	Malik Delacruz	Male	United States	580-7613 Nunc Road	Tallahassee	Florida	32301	dolor.Fusce@Mauris.co.uk	8	2/20/16 9:24
78315977299	Charitv Landlev	Female	United States	109-7307 Cras Road	Houston	Texas	77001	Donec@Nuncmauris.com	8	6/10/16 16:16
87972646599	Jocelvyn Havden	Female	United States	8019 Eu Rd.	Santa Clara	California	95051	maona.a.tortor@sitametluctus.ora	16	5/20/16 17:24
70897853699	Justine Gates	Female	United States	Ao #647-2055 Malesuad...	Seattle	Washington	98105	at.libero@luctusetultrices.edu	6	12/10/16 12:34
23547986539	Julia Wano	Female	United States	2500 Broadwav	Lubbock	Texas	79409	Julia.W@gmail.com	5	08/20/17 23:10
Last_Visit	Order_id	Product_id	Price	Order_quantity	Coupon_ID	Discount_Value	payment	Order_status		
0001-04-18 11:23:00	204	2	108	1	A3	0.05	102.6	delivered		
0003-03-18 11:22:00	237	4	165	1	A3	0.05	156.75	delivered		
0002-03-18 23:21:00	245	5	95	2	B4	0.2	152	delivered		
0001-10-18 01:26:00	222	8	123	1	B6	0.1	110.7	delivered		
0001-04-18 11:23:00	249	8	123	1	B6	0.1	110.7	delivered		
0003-03-18 11:22:00	246	1	80	2	A6	0.15	136	delivered		
0001-09-18 08:15:00	202	2	108	2	B7	0.2	172.8	delivered		
0002-03-18 23:21:00	212	5	95	1	B8	0.2	76	delivered		
0001-03-18 02:33:00	230	5	95	2	B8	0.2	152	delivered		
0003-03-18 11:22:00	213	6	64	1	A8	0.3	44.8	delivered		
0002-06-18 23:56:00	224	6	64	4	A8	0.3	179.2	shipped		
0002-03-18 23:21:00	236	6	64	1	A8	0.3	44.8	delivered		
0001-02-18 13:35:00	206	4	165	1	NULL	NULL	165	delivered		
0003-05-18 10:40:00	208	9	71	2	NULL	NULL	142	delivered		
0003-05-18 10:40:00	208	6	64	1	NULL	NULL	64	delivered		
0003-02-18 10:31:00	217	2	108	1	NULL	NULL	108	delivered		
0001-03-18 02:33:00	218	1	80	1	NULL	NULL	80	delivered		
0002-07-18 01:14:00	221	1	80	1	NULL	NULL	80	delivered		
0001-04-18 11:23:00	225	5	95	2	NULL	NULL	190	delivered		
0003-05-18 10:40:00	229	9	71	1	NULL	NULL	71	failed		
0002-07-18 01:14:00	233	5	95	2	NULL	NULL	190	delivered		
0001-10-18 01:26:00	234	4	165	2	NULL	NULL	330	delivered		
0003-03-18 11:22:00	237	2	108	1	NULL	NULL	108	delivered		
0003-05-18 10:40:00	241	10	88	1	NULL	NULL	88	delivered		
2010-12-17 23:11:00	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL		

Triggers

1. Trigger for preventing from inserting same customer information

DELIMITER \$\$

CREATE TRIGGER NO_DULICATE_CUSTOMER_TRIGGER

BEFORE INSERT ON customer

FOR EACH ROW

BEGIN

IF EXISTS (SELECT * FROM customer c WHERE c.Customer_Name =
NEW.Customer_Name and c.ZIP = NEW.ZIP) THEN

SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Warning: This customer
already exists';

END IF;

END\$\$

DELIMITER ;

2. Trigger for coupon expiration setting

DELIMITER \$\$

CREATE TRIGGER COUPON_EXPIRATION_SETTING_TRIGGER


```

BEFORE INSERT ON coupon
FOR EACH ROW
BEGIN
    IF NEW.Expiration_Time < NOW()
    THEN
        SIGNAL SQLSTATE '02000' SET MESSAGE_TEXT = 'Warning: Expiration date can
not be earlier than now!';
    END IF;
END$$
DELIMITER ;

```

```

3. Trigger for notifying expired coupon
DELIMITER $$

CREATE TRIGGER NO_EXPIRATION_COUPON_TRIGGER
BEFORE INSERT ON order_product_info
FOR EACH ROW
BEGIN
    IF (SELECT * FROM `order_product_info` op join coupon c
    ON c.Coupon_ID = op.Coupon_ID where c.Expiration_Time < NOW() )
    THEN
        SIGNAL SQLSTATE '02000' SET MESSAGE_TEXT = 'Warning: This coupon has been
expired!';
    END IF;
END$$
DELIMITER ;

```

Metrics

1. Total sales amount change by product which was bought with coupon compared to without coupon?

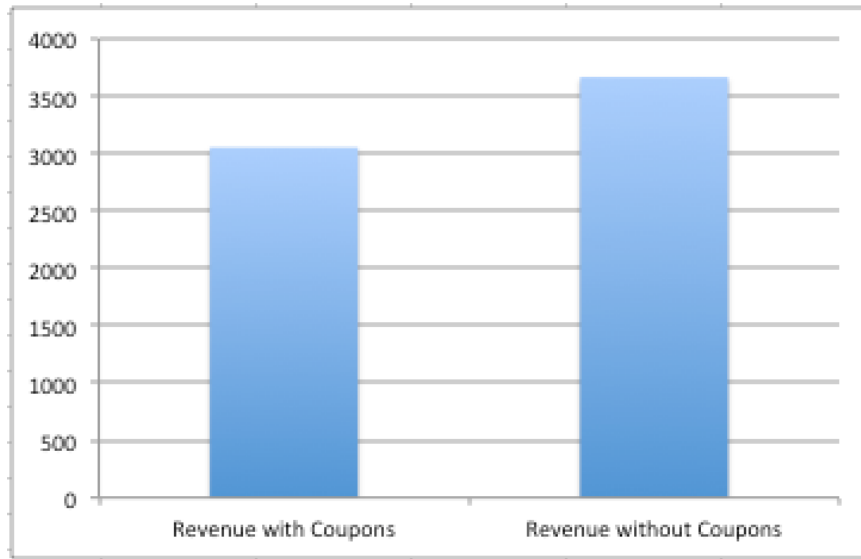
```
SELECT sum(n.Price*(1-c.Discount_Value)*n.Order_quantity)
       as 'Revenue with Coupons'
from (select p.Price, o.Order_quantity, o.coupon_id
from order_product_info o
left join product p
on p.Product_id = o.Product_id
WHERE o.coupon_id IS not NULL) n
left join coupon c
on n.coupon_id = c.Coupon_ID;
```

Revenue with Coupons
3052.4000000000005

```
SELECT sum(p.Price*o.Order_quantity) as 'Revenue without Coupons'
from order_product_info o
left join product p
on p.Product_id = o.Product_id
WHERE o.coupon_id IS NULL;
```

Revenue without Coupons

3665



2. Number of sales with coupons vs. without coupons? By product?

```
SELECT COUNT(distinct o.order_id) AS `Number of Orders w/Coupons` FROM  
`order` o JOIN order_product_info opi  
ON o.order_id = opi.order_id  
WHERE coupon_id IS NOT NULL;
```

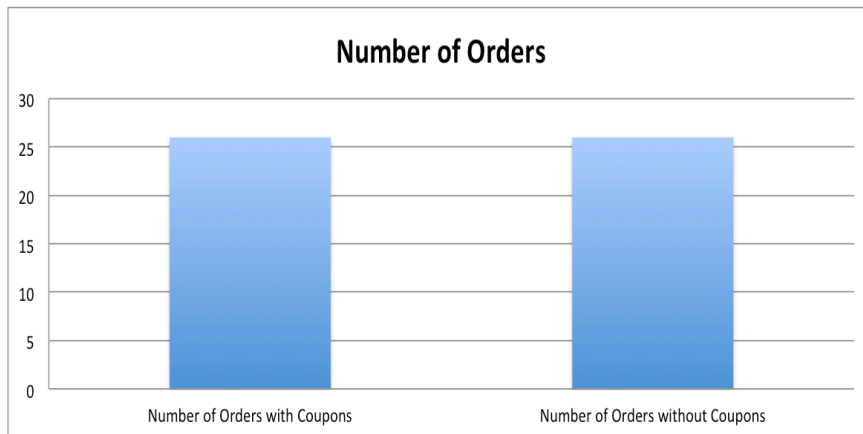
Number of Orders w/Coupons

26

```
SELECT COUNT(distinct o.order_id) AS `Number of Orders without Coupons`  
FROM `order` o JOIN order_product_info opi  
ON o.order_id = opi.order_id  
WHERE coupon_id IS NULL;
```

Number of Orders without Coupons

26



```
select product_id, count(distinct o.order_id) as `Number of Orders w/o Coupons`  
from `order` o join order_product_info p  
on o.order_id = p.order_id  
where coupon_id is null  
group by product_id;
```

product_id	Number of Orders w/o Coupons
1	3
2	3
3	4
4	3
5	4
6	1
7	4
8	1
9	3
10	3

```
select product_id, count(distinct o.order_id) as `Number of Orders w/Coupons`  
from `order` o join order_product_info p
```

on o.order_id = p.order_id

where coupon_id is not null

group by product_id;

product_id	Number of Orders w/Coupons
1	3
2	2
3	3
4	1
5	7
6	4
8	6



3. Which product types are our bestsellers (most ordered)? Which product type was the least popular (least ordered)?

```
SELECT category_name, SUM(order_quantity) AS `Quantity Ordered` FROM  
order_product_info p JOIN product pr
```

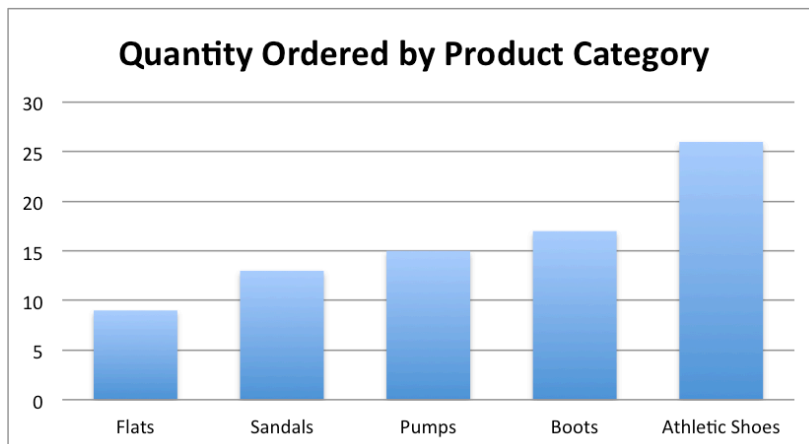
```
ON pr.product_id = p.product_id
```

```
JOIN product_category c ON c.category_id = pr.category_id
```

```
GROUP BY category_name
```

```
ORDER BY `Quantity Ordered` DESC;
```

category_name	Quantity Ordered
Athletic shoes	26
Boots	17
Pumps	15
Sandals	13
Flats	9



4. How many orders each month?

```
SELECT month(Order_date) as month, count(*) as order_number_permonth
```

```
FROM `order`
```

```
GROUP BY month(Order_date);
```

month	order_number_permonth
1	2
2	6
3	2
4	7
5	1
6	4
7	3
8	3
9	7
10	2
11	3
12	11

