Database Project Report

Elif Keleş - 161044033

May 29th, 2021

In this assignment, I was asked to create a company's database. I decided to create my company as an online shopping store. This document consists of the user requirements, E-R diagram, functional dependency and normalization data and the table sections of my company database called myShop. To create my company's database I used MySQL DBMS. Then I designed my shop and filled its tables with related random data.

1 User Requirements

myShop is designed as an online shopping site. It has branches, employees, customers and products.

1.1 Employee

Employees work at myShop. They have thier unique IDs. And they have a manager. A manager is also an employee, who runs one of the branches.

1.2 Branch

There are three branches of myShop. Every branch has their unique ID and manager.

1.3 Customer

Everyone who shop from myShop is a customer. They need to provide their phone numbers, addresses, names and birthdays to shop. Every customer has a unique ID, and a total shopping value which keeps the data of number of times the customer makes a purchase. They can order a product.

1.4 Products

Every brand has some products to sell on myShop. Products have price and quantity numbers. Also they have a category value.

1.5 Orders

After an order is made, the customer gets an invoice from myShop which shows the total price of the shopping and customer's and order's IDs. Every order has a status value. Status is 'preparing' as default. If order is ready, the status changes to 'on the way'. And when it is arrived it changes to 'delivered'.

2 E-R Diagram of the Company

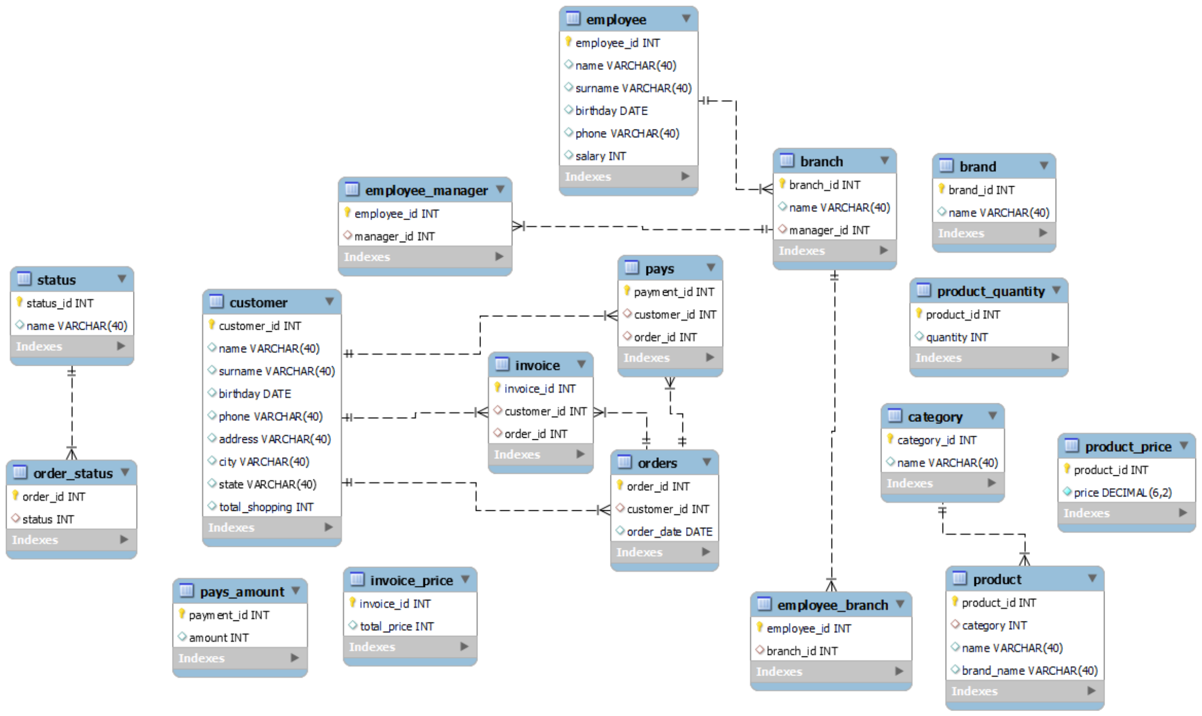


Figure 1: E-R Diagram

3 Normalization

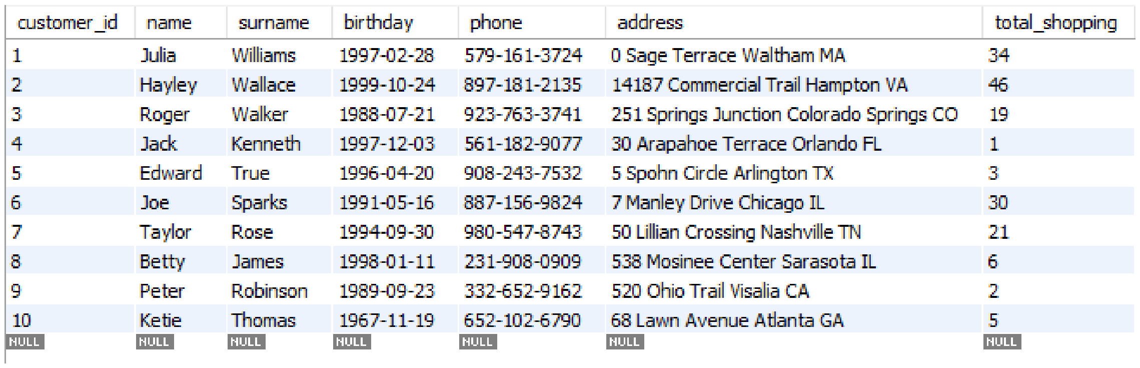
Normalization is a technique of organizing data into multiple related tables, while reducing data redundancy. We use normalisation because we want to make our database tables as useful as possible and we want to access data fast. If we don't normalize the tables we might face update, insert and delete problems.

3.1 First Normal Form

- Rule 1) Each column should contain only one data (atomic values).
- Rule 2) Each column should have the same type of values.
- Rule 3) Each column should have a unique name.

- Rule 4) Order of data does not matter.

I paid attention to these rules when creating my tables. The only change was on customer table's address attributes. After normalization address attribute is separated into three attributes as address, city, state.



customer_id	name	surname	birthday	phone	address	total_shopping
1	Julia	Williams	1997-02-28	579-161-3724	0 Sage Terrace Waltham MA	34
2	Hayley	Wallace	1999-10-24	897-181-2135	14187 Commercial Trail Hampton VA	46
3	Roger	Walker	1988-07-21	923-763-3741	251 Springs Junction Colorado Springs CO	19
4	Jack	Kenneth	1997-12-03	561-182-9077	30 Arapahoe Terrace Orlando FL	1
5	Edward	True	1996-04-20	908-243-7532	5 Spohn Circle Arlington TX	3
6	Joe	Sparks	1991-05-16	887-156-9824	7 Manley Drive Chicago IL	30
7	Taylor	Rose	1994-09-30	980-547-8743	50 Lillian Crossing Nashville TN	21
8	Betty	James	1998-01-11	231-908-0909	538 Mosinee Center Sarasota IL	6
9	Peter	Robinson	1989-09-23	332-652-9162	520 Ohio Trail Visalia CA	2
10	Ketie	Thomas	1967-11-19	652-102-6790	68 Lawn Avenue Atlanta GA	5
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Figure 2: Customer table before 1st NF

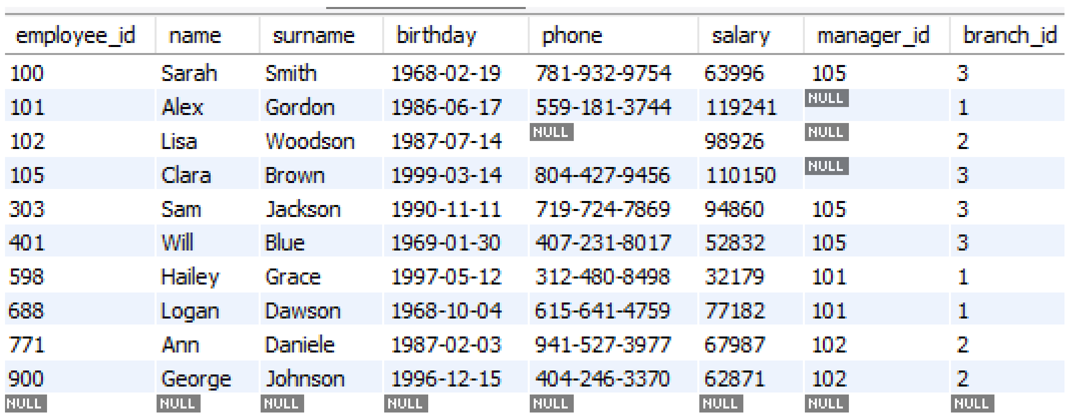
3.2 Second Normal Form

- Rule 1) The tables should be in the 1st Normal Form.
- Rule 2) There should be no partial dependencies.

To achieve second NF, I found all the functional dependencies first. After I found the super keys, I checked the candidate keys with closure. Then, I found, prime and non-prime attributes to check partial dependencies.

If every attribute of the table is prime attribute, then it is in 2nd Normal Form. Otherwise, we have to make decomposition(lossless).

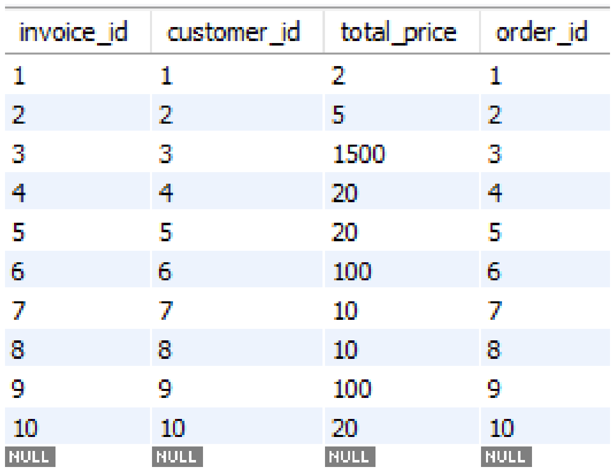
The tables I have decompsed are:



employee_id	name	surname	birthday	phone	salary	manager_id	branch_id
100	Sarah	Smith	1968-02-19	781-932-9754	63996	105	3
101	Alex	Gordon	1986-06-17	559-181-3744	119241	NULL	1
102	Lisa	Woodson	1987-07-14	NULL	98926	NULL	2
105	Clara	Brown	1999-03-14	804-427-9456	110150	NULL	3
303	Sam	Jackson	1990-11-11	719-724-7869	94860	105	3
401	Will	Blue	1969-01-30	407-231-8017	52832	105	3
598	Hailey	Grace	1997-05-12	312-480-8498	32179	101	1
688	Logan	Dawson	1968-10-04	615-641-4759	77182	101	1
771	Ann	Daniele	1987-02-03	941-527-3977	67987	102	2
900	George	Johnson	1996-12-15	404-246-3370	62871	102	2
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Figure 3: Employee table before 2nd NF

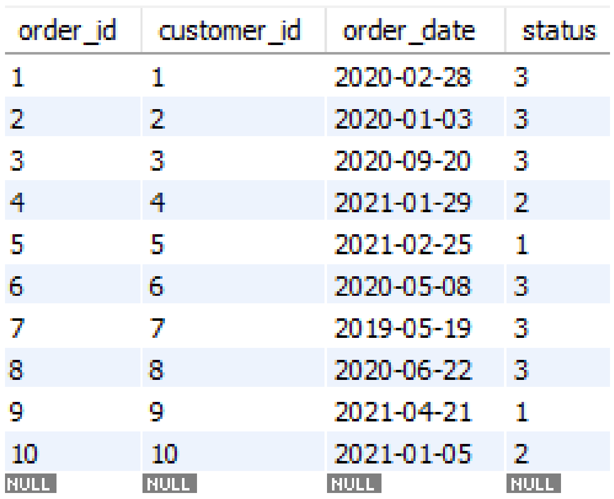
Employee table is separated into three tables as employee, employee_branch and employee_manager.



invoice_id	customer_id	total_price	order_id
1	1	2	1
2	2	5	2
3	3	1500	3
4	4	20	4
5	5	20	5
6	6	100	6
7	7	10	7
8	8	10	8
9	9	100	9
10	10	20	10
NULL	NULL	NULL	NULL

Figure 4: Invoice table before 2nd NF

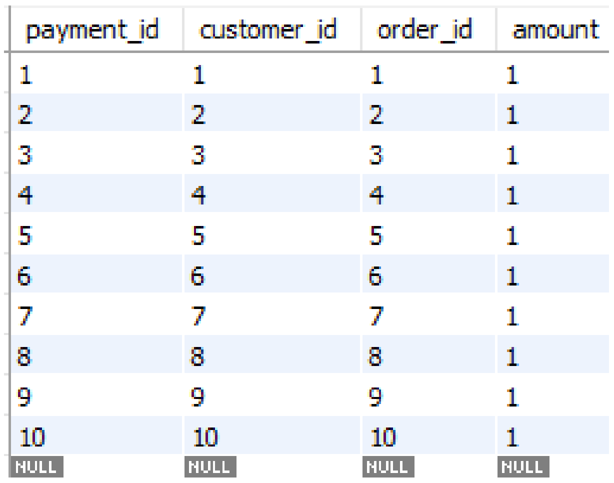
Invoice table is separated into two tables as invoice and invoice_price.



order_id	customer_id	order_date	status
1	1	2020-02-28	3
2	2	2020-01-03	3
3	3	2020-09-20	3
4	4	2021-01-29	2
5	5	2021-02-25	1
6	6	2020-05-08	3
7	7	2019-05-19	3
8	8	2020-06-22	3
9	9	2021-04-21	1
10	10	2021-01-05	2
NULL	NULL	NULL	NULL

Figure 5: Orders table before 2nd NF

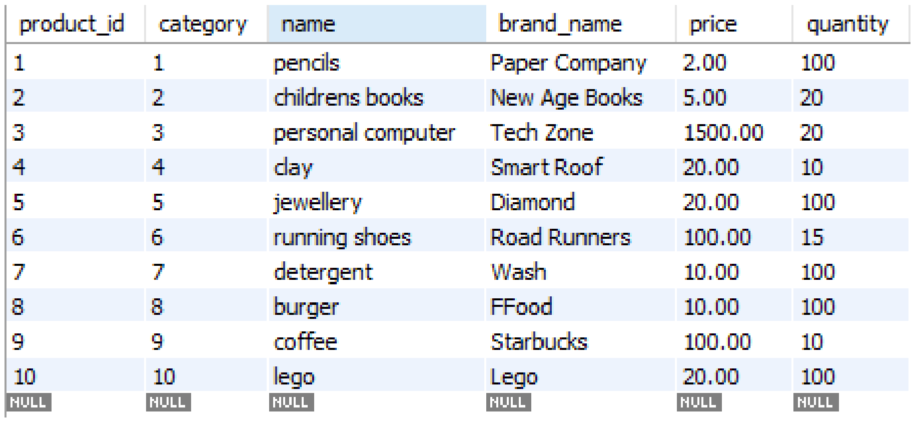
Orders table is separated into two as orders and orders_status table.



payment_id	customer_id	order_id	amount
1	1	1	1
2	2	2	1
3	3	3	1
4	4	4	1
5	5	5	1
6	6	6	1
7	7	7	1
8	8	8	1
9	9	9	1
10	10	10	1
NULL	NULL	NULL	NULL

Figure 6: Payment table before 2nd NF

Payment table is separated into two as payment and payment_amount table.



product_id	category	name	brand_name	price	quantity
1	1	pencils	Paper Company	2.00	100
2	2	childrens books	New Age Books	5.00	20
3	3	personal computer	Tech Zone	1500.00	20
4	4	clay	Smart Roof	20.00	10
5	5	jewellery	Diamond	20.00	100
6	6	running shoes	Road Runners	100.00	15
7	7	detergent	Wash	10.00	100
8	8	burger	FFood	10.00	100
9	9	coffee	Starbucks	100.00	10
10	10	lego	Lego	20.00	100
NULL	NULL	NULL	NULL	NULL	NULL

Figure 7: Product table before 2nd NF

Product table is separated into three tables as product, product_price, product_quantity.

3.3 Third Normal Form

- Rule 1) The tables should be in the 2nd Normal Form.
- Rule 2) They should have no Transitive Dependency.(for non-prime attributes)

3.4 BCNF

- Rule 1) The tables should be in the 3rd Normal Form.
- Rule 2) For any dependency A->B, A should be a super key.

Since, for all of my tables, the left hand side attributes (of relations) are super keys, I did not change any of the tables in these two steps.

4 Functional Dependencies

For any two tuples x and y, we should check the functional dependency.
x (determinant) -> y (dependent)
For functional dependency: If the determinants are equal, then the dependents must be equal, as well.

Since I used unique id values for each table, I can use them as primary key.

4.1 For Branch

- branch_id -> name
- branch_id -> manager_id
- branch_id, name -> manager_id
- branch_id, manager_id -> name
- name -> branch_id
- name -> manager_id
- name, branch_id -> manager_id
- name, manager_id -> branch_id
- manager_id -> branch_id
- manager_id -> name
- manager_id, name -> branch_id
- manager_id, branch_id -> name

Click 8 for branch table

4.2 For Brand

- brand_id -> name
- name -> brand_id

Click 9 for brand table

4.3 For Category

- category_id -> name
- name -> category_id

Click 10 for category table

4.4 For Customer

- customer_id -> name
- customer_id -> surname
- customer_id -> birthday
- customer_id -> phone
- customer_id -> address
- customer_id -> city
- customer_id -> state
- customer_id -> total_shopping
- Every other dependency starting with customer_id
- name -> customer_id
- name -> surname
- name -> birthday
- name -> phone
- name -> address
- name -> city
- name -> state
- name -> total_shopping
- Every other dependency starting with name
- surname -> customer_id
- surname -> name
- surname -> birthday
- surname -> phone
- surname -> address
- surname -> city
- surname -> state

- surname -> total_shopping
- Every other dependency starting with surname
- birthday -> customer_id
- birthday -> name
- birthday -> surname
- birthday -> phone
- birthday -> address
- birthday -> city
- birthday -> state
- birthday -> total_shopping
- Every other dependency starting with birthday
- phone -> customer_id
- phone -> surname
- phone -> birthday
- phone -> name
- phone -> address
- phone -> city
- phone -> state
- phone -> total_shopping
- Every other dependency starting with phone
- address -> customer_id
- address -> surname
- address -> birthday
- address -> phone
- address -> name
- address -> city
- address -> state
- address -> total_shopping
- Every other dependency starting with address
- city -> customer_id
- city -> surname
- city -> birthday

- city -> phone
- city -> address
- city -> name
- city -> state
- city -> total_shopping
- Every other dependency starting with city
- state -> customer_id
- state -> surname
- state -> birthday
- state -> phone
- state -> address
- state -> city
- state -> name
- state -> total_shopping
- Every other dependency starting with state
- total_shopping -> customer_id
- total_shopping -> surname
- total_shopping -> birthday
- total_shopping -> phone
- total_shopping -> address
- total_shopping -> city
- total_shopping -> state
- total_shopping -> name
- Every other dependency starting with total_shopping

Click 11 for customer table

4.5 For Employee

- employee_id -> name
- employee_id -> surname
- employee_id -> birthday
- employee_id -> phone
- employee_id -> salary
- Every other dependency starting with employee_id
- name -> employee_id
- name -> surname
- name -> birthday
- name -> phone
- name -> salary
- Every other dependency starting with name
- surname -> employee_id
- surname -> name
- surname -> birthday
- surname -> phone
- surname -> salary
- Every other dependency starting with surname
- birthday -> employee_id
- birthday -> surname
- birthday -> name
- birthday -> phone
- birthday -> salary
- Every other dependency starting with birthday
- phone -> employee_id
- phone -> surname
- phone -> birthday
- phone -> name
- phone -> salary
- Every other dependency starting with phone

- salary -> employee_id
- salary -> surname
- salary -> birthday
- salary -> phone
- salary -> name
- Every other dependency starting with salary

Click 12 for employee table

4.6 For Employee_Branch

- employee_id -> branch_id

Click 13 for employee_branch table

4.7 For Employee_Manager

- employee_id -> manager_id

Click 14 for employee_manager table

4.8 For Invoice

- invoice_id -> customer_id
- invoice_id -> order_id
- invoice_id, customer_id -> order_id
- invoice_id, order_id -> customer_id
- customer_id -> invoice_id
- customer_id -> total_price
- customer_id -> order_id
- customer_id, invoice_id -> order_id
- customer_id, order_id -> invoice_id
- order_id -> invoice_id
- order_id -> customer_id
- order_id, invoice_id -> customer_id
- order_id, customer_id -> invoice_id

Click 15 for invoice table

4.9 For Invoice_Price

- invoice_id -> total_price

Click 16 for invoice_price table

4.10 For Orders

- order_id -> customer_id
- order_id -> order_date
- Every other dependency starting with order_id
- customer_id -> order_id
- customer_id -> order_date
- Every other dependency starting with customer_id
- order_date -> order_id
- order_date -> customer_id
- Every other dependency starting with order_date

Click 17 for orders table

4.11 For Orders_Status

- order_id -> status

Click 18 for orders_status table

4.12 For Payment

- payment_id -> customer_id
- payment_id -> order_id
- Every other dependency starting with payment_id
- customer_id -> payment_id
- customer_id -> order_id
- Every other dependency starting with customer_id
- order_id -> payment_id
- order_id -> customer_id
- Every other dependency starting with order_id

Click 19 for payment table

4.13 For Payment_Amount

- payment_id -> amount

Click 20 for payment_amount table

4.14 For Product

- product_id -> category
- product_id -> name
- product_id -> brand_name
- Every other dependency starting with product_id
- category -> product_id
- category -> name
- category -> brand_name
- Every other dependency starting with category
- name -> product_id
- name -> category
- name -> brand_name
- Every other dependency starting with name
- brand_name -> product_id
- brand_name -> name
- brand_name -> category
- Every other dependency starting with brand_name

Click 21 for product table

4.15 For Product_Price

- product_id -> price

Click 22 for product_price table

4.16 For Product_Quantity

- product_id -> quantity

Click 23 for product_quantity table

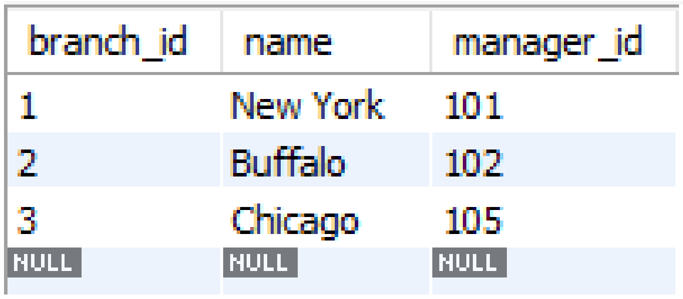
4.17 For Status

- status_id -> name
- name -> status_id

Click 42 for status table

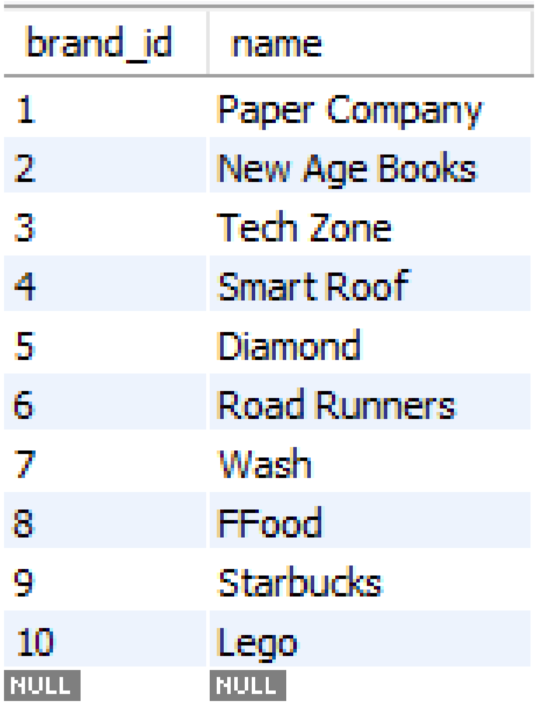
** I simplified some of the dependencies, to reduce complication.

5 List of the Tables



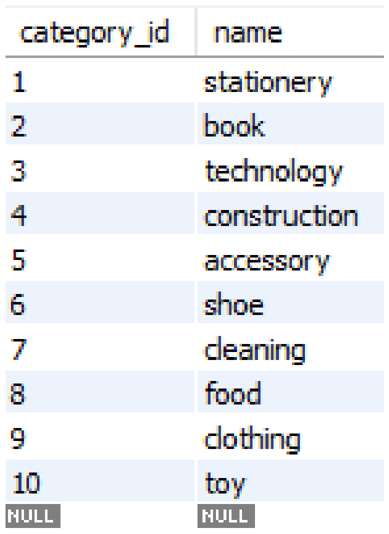
branch_id	name	manager_id
1	New York	101
2	Buffalo	102
3	Chicago	105
NULL	NULL	NULL

Figure 8: Branch Table



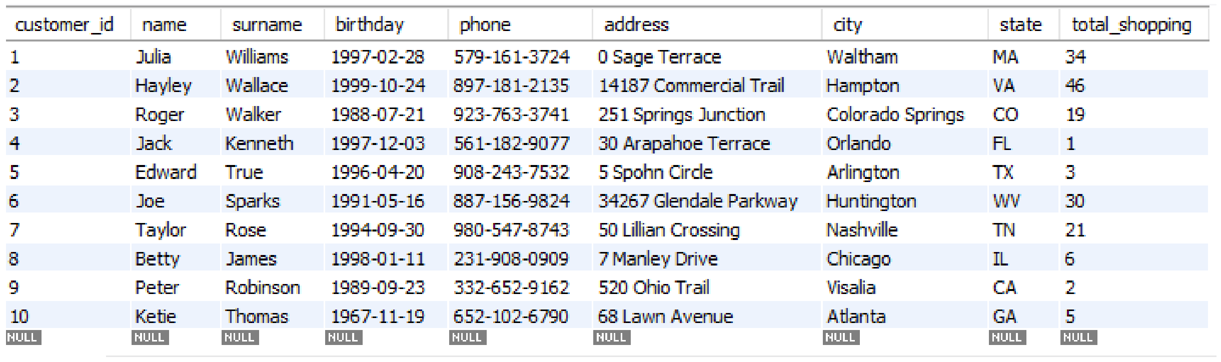
brand_id	name
1	Paper Company
2	New Age Books
3	Tech Zone
4	Smart Roof
5	Diamond
6	Road Runners
7	Wash
8	FFood
9	Starbucks
10	Lego
NULL	NULL

Figure 9: Brand Table



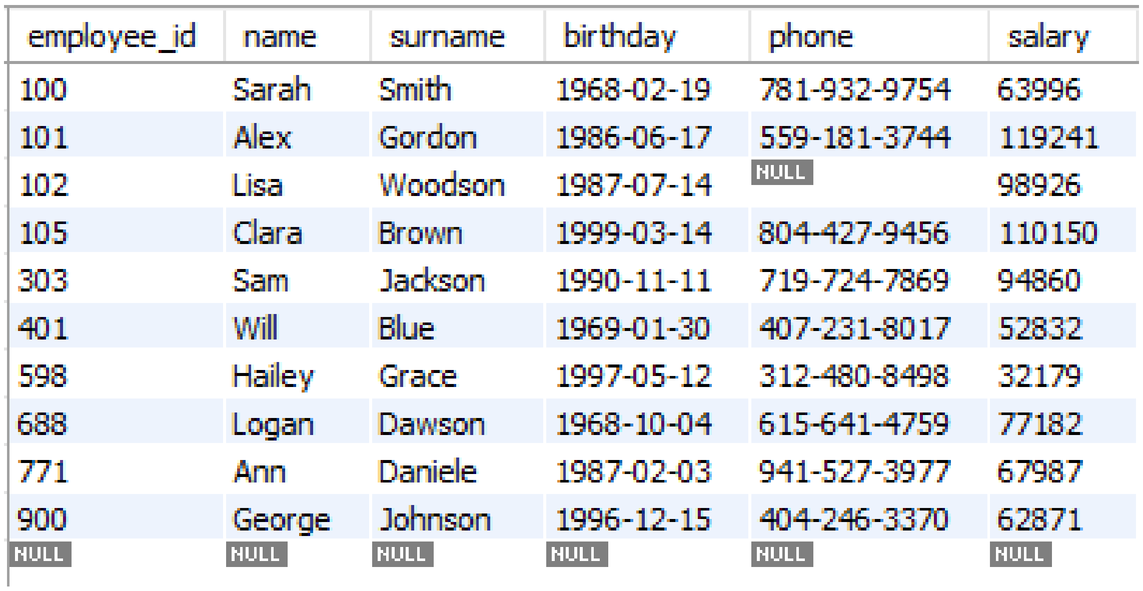
category_id	name
1	stationery
2	book
3	technology
4	construction
5	accessory
6	shoe
7	cleaning
8	food
9	clothing
10	toy
NULL	NULL

Figure 10: Category Table



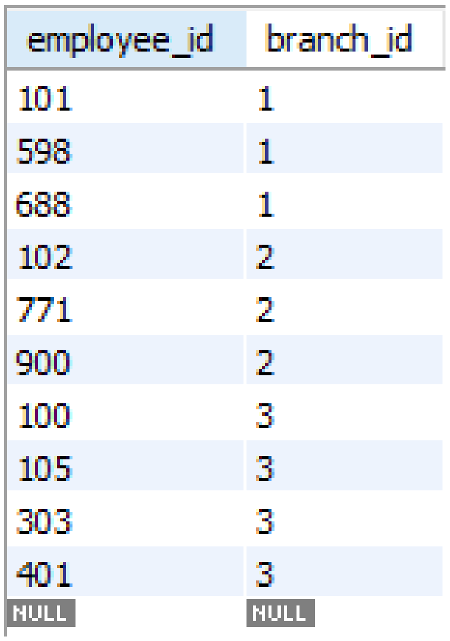
customer_id	name	surname	birthday	phone	address	city	state	total_shopping
1	Julia	Williams	1997-02-28	579-161-3724	0 Sage Terrace	Waltham	MA	34
2	Hayley	Wallace	1999-10-24	897-181-2135	14187 Commercial Trail	Hampton	VA	46
3	Roger	Walker	1988-07-21	923-763-3741	251 Springs Junction	Colorado Springs	CO	19
4	Jack	Kenneth	1997-12-03	561-182-9077	30 Arapahoe Terrace	Orlando	FL	1
5	Edward	True	1996-04-20	908-243-7532	5 Spohn Circle	Arlington	TX	3
6	Joe	Sparks	1991-05-16	887-156-9824	34267 Glendale Parkway	Huntington	WV	30
7	Taylor	Rose	1994-09-30	980-547-8743	50 Lillian Crossing	Nashville	TN	21
8	Betty	James	1998-01-11	231-908-0909	7 Manley Drive	Chicago	IL	6
9	Peter	Robinson	1989-09-23	332-652-9162	520 Ohio Trail	Visalia	CA	2
10	Ketie	Thomas	1967-11-19	652-102-6790	68 Lawn Avenue	Atlanta	GA	5
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Figure 11: Customer Table



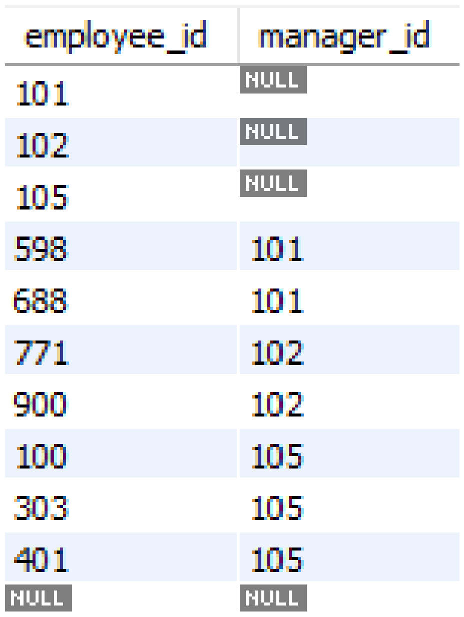
employee_id	name	surname	birthday	phone	salary
100	Sarah	Smith	1968-02-19	781-932-9754	63996
101	Alex	Gordon	1986-06-17	559-181-3744	119241
102	Lisa	Woodson	1987-07-14	NULL	98926
105	Clara	Brown	1999-03-14	804-427-9456	110150
303	Sam	Jackson	1990-11-11	719-724-7869	94860
401	Will	Blue	1969-01-30	407-231-8017	52832
598	Hailey	Grace	1997-05-12	312-480-8498	32179
688	Logan	Dawson	1968-10-04	615-641-4759	77182
771	Ann	Daniele	1987-02-03	941-527-3977	67987
900	George	Johnson	1996-12-15	404-246-3370	62871
NULL	NULL	NULL	NULL	NULL	NULL

Figure 12: Employee Table



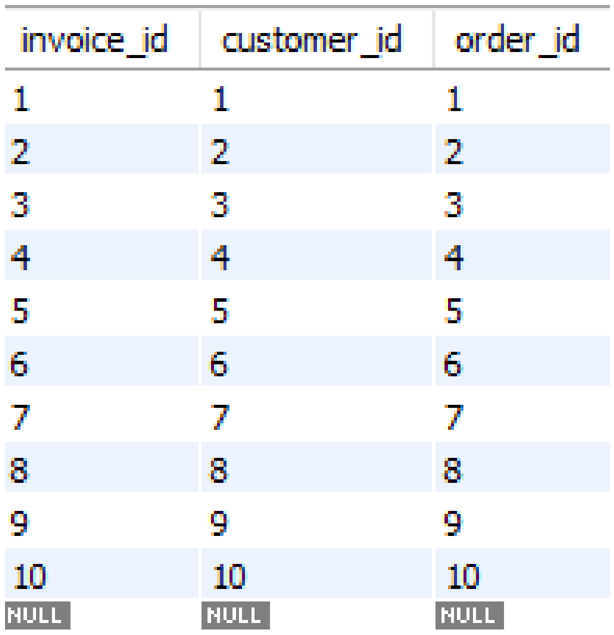
employee_id	branch_id
101	1
598	1
688	1
102	2
771	2
900	2
100	3
105	3
303	3
401	3
NULL	NULL

Figure 13: Employee_Branch Table



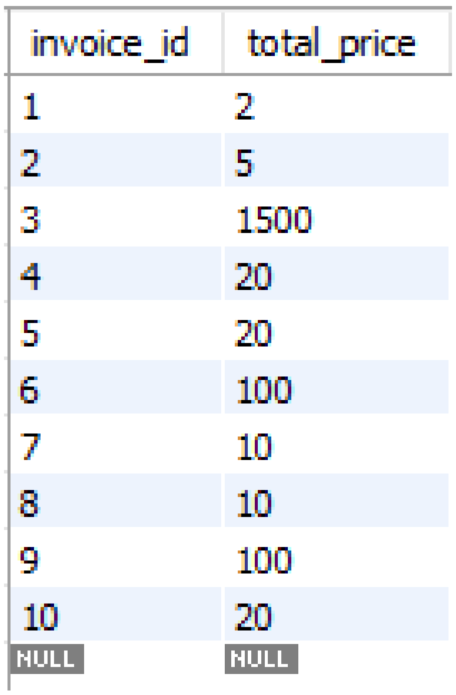
employee_id	manager_id
101	NULL
102	NULL
105	NULL
598	101
688	101
771	102
900	102
100	105
303	105
401	105
NULL	NULL

Figure 14: Employee_Manager Table



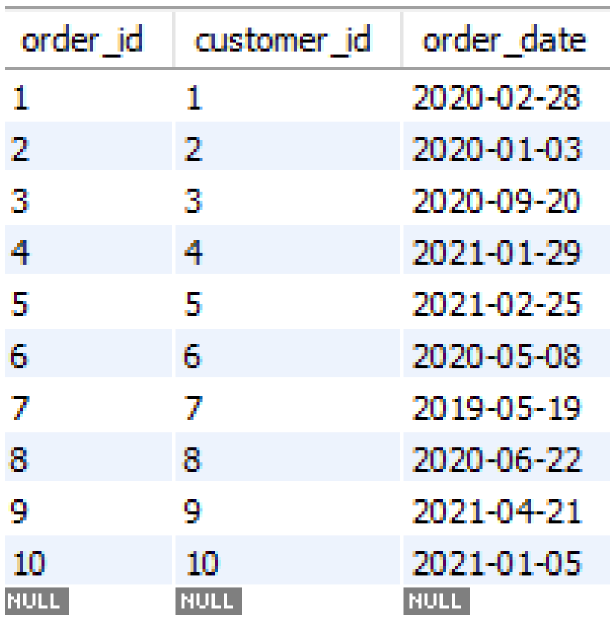
invoice_id	customer_id	order_id
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5
6	6	6
7	7	7
8	8	8
9	9	9
10	10	10
NULL	NULL	NULL

Figure 15: Invoice Table



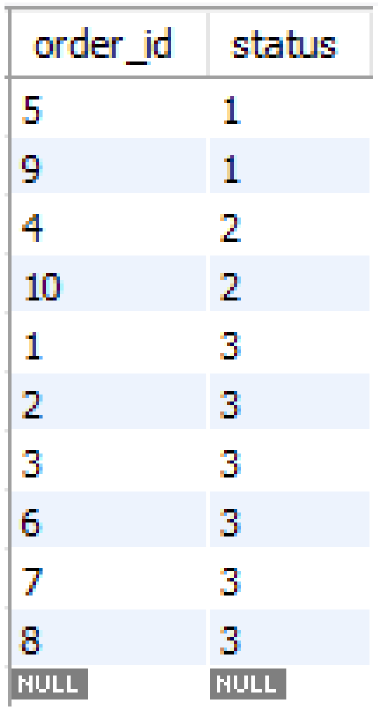
invoice_id	total_price
1	2
2	5
3	1500
4	20
5	20
6	100
7	10
8	10
9	100
10	20
NULL	NULL

Figure 16: Invoice_Price Table



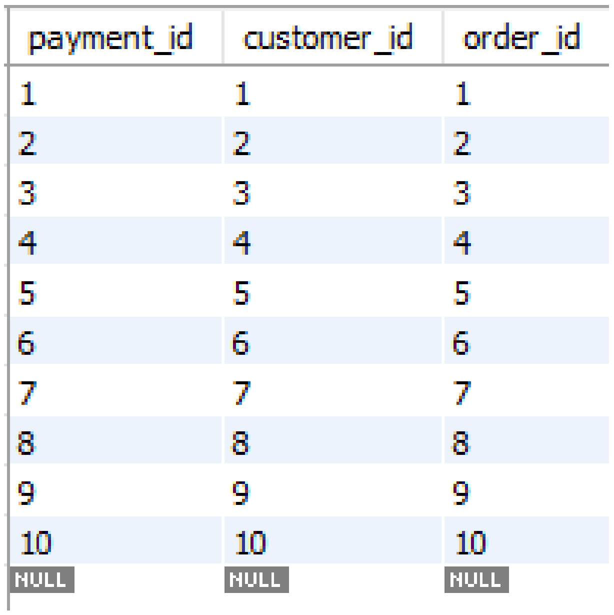
order_id	customer_id	order_date
1	1	2020-02-28
2	2	2020-01-03
3	3	2020-09-20
4	4	2021-01-29
5	5	2021-02-25
6	6	2020-05-08
7	7	2019-05-19
8	8	2020-06-22
9	9	2021-04-21
10	10	2021-01-05
NULL	NULL	NULL

Figure 17: Orders Table



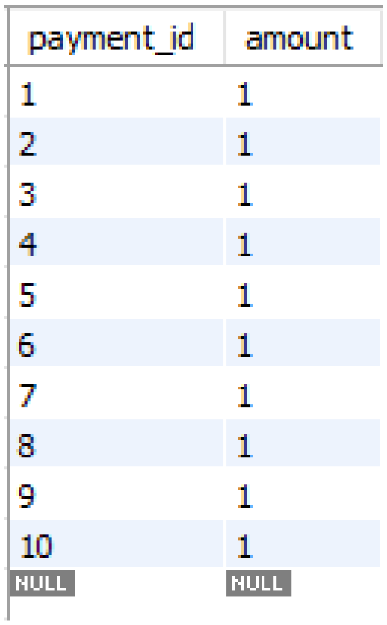
order_id	status
5	1
9	1
4	2
10	2
1	3
2	3
3	3
6	3
7	3
8	3
NULL	NULL

Figure 18: Orders_Status Table



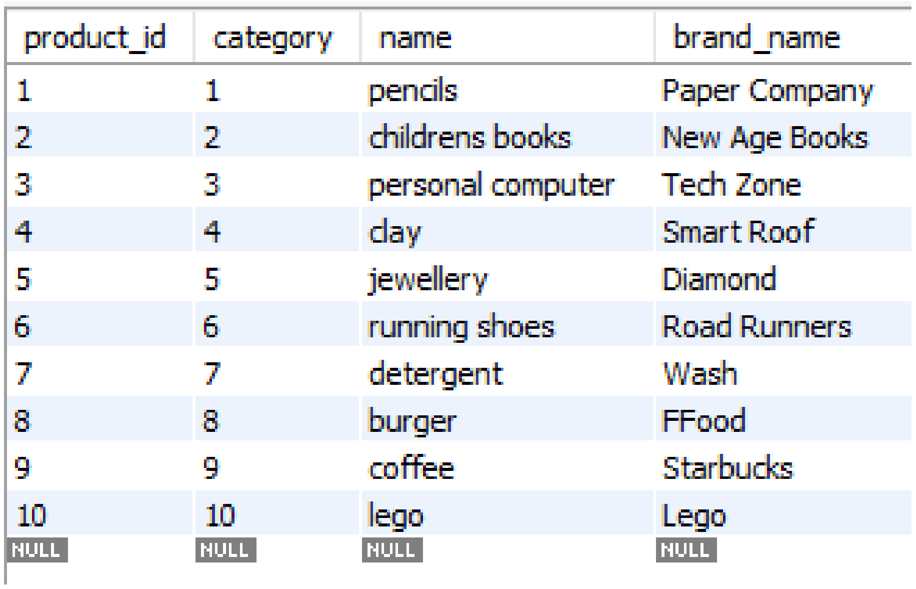
payment_id	customer_id	order_id
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5
6	6	6
7	7	7
8	8	8
9	9	9
10	10	10
NULL	NULL	NULL

Figure 19: Payment Table



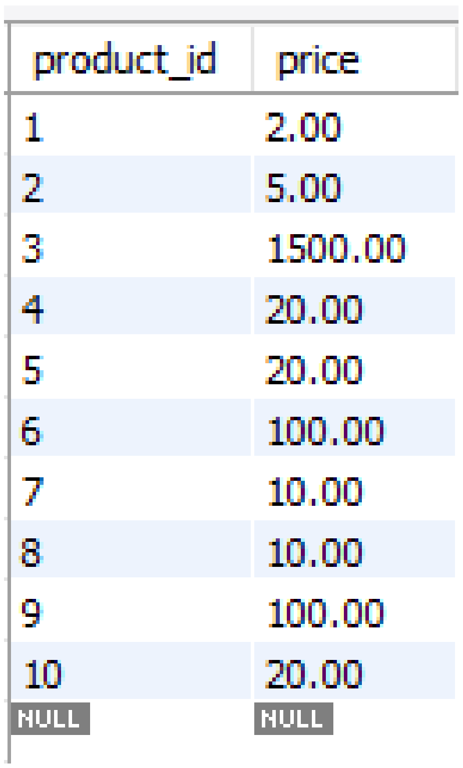
payment_id	amount
1	1
2	1
3	1
4	1
5	1
6	1
7	1
8	1
9	1
10	1
NULL	NULL

Figure 20: Payment_Amount Table



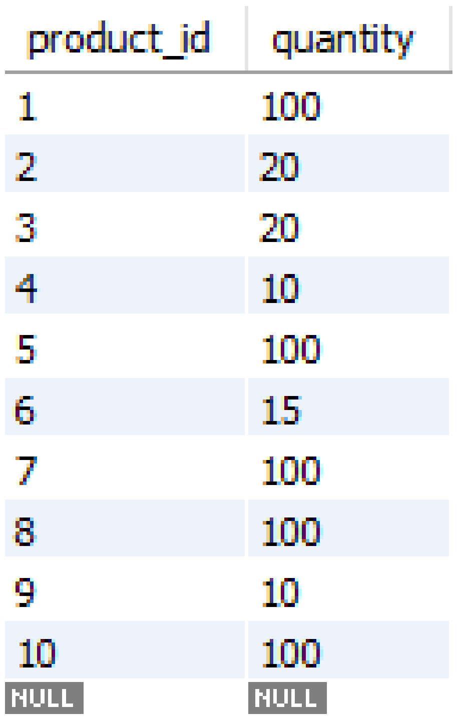
product_id	category	name	brand_name
1	1	pencils	Paper Company
2	2	childrens books	New Age Books
3	3	personal computer	Tech Zone
4	4	clay	Smart Roof
5	5	jewellery	Diamond
6	6	running shoes	Road Runners
7	7	detergent	Wash
8	8	burger	FFood
9	9	coffee	Starbucks
10	10	lego	Lego
NULL	NULL	NULL	NULL

Figure 21: Product Table



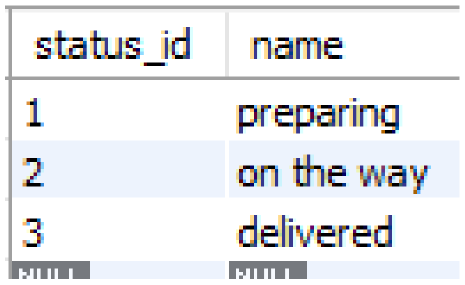
product_id	price
1	2.00
2	5.00
3	1500.00
4	20.00
5	20.00
6	100.00
7	10.00
8	10.00
9	100.00
10	20.00
NULL	NULL

Figure 22: Product_Price Table



product_id	quantity
1	100
2	20
3	20
4	10
5	100
6	15
7	100
8	100
9	10
10	100
NULL	NULL

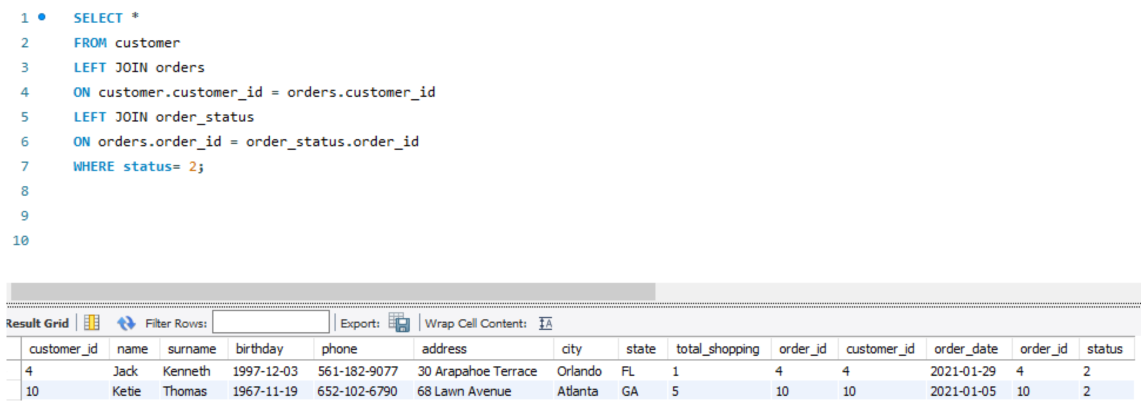
Figure 23: Product_Quantity Table



status_id	name
1	preparing
2	on the way
3	delivered
NULL	NULL

Figure 24: Status Table

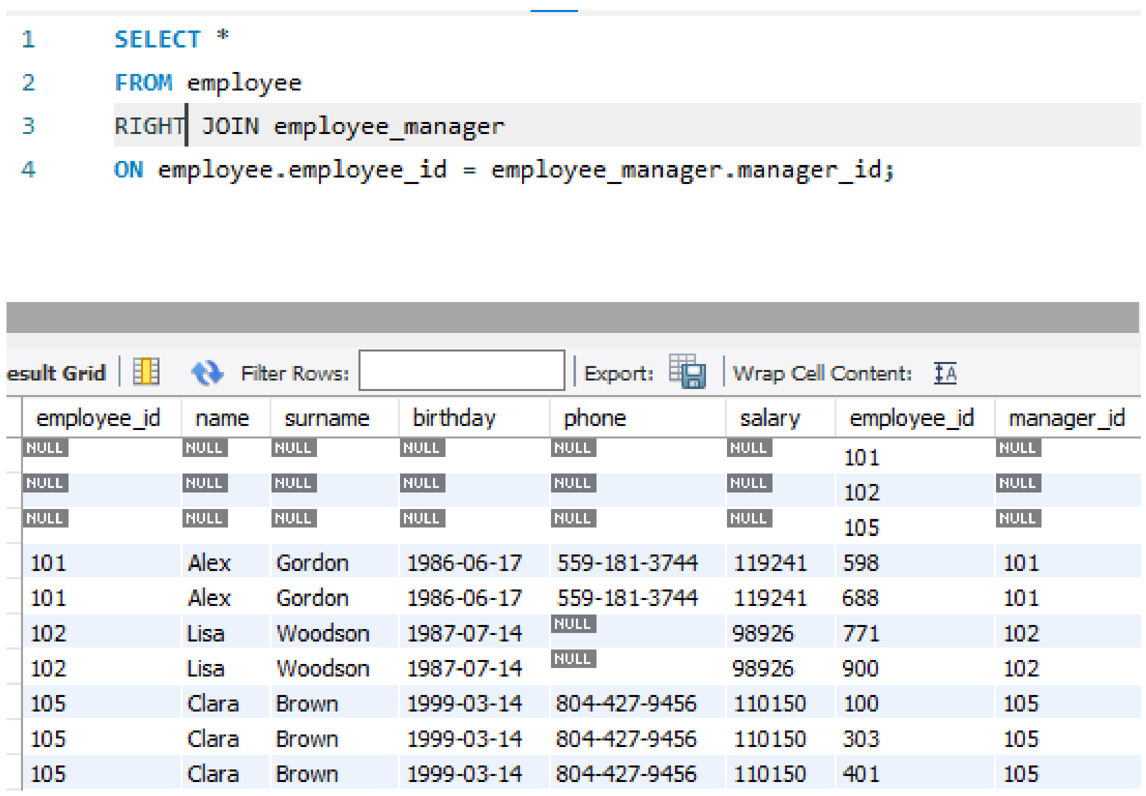
6 Join Queries



```sql
SELECT *
FROM customer
LEFT JOIN orders
ON customer.customer_id = orders.customer_id
LEFT JOIN order_status
ON orders.order_id = order_status.order_id
WHERE status= 2;
```

customer_id	name	surname	birthday	phone	address	city	state	total_shopping	order_id	customer_id	order_date	order_id	status
4	Jack	Kenneth	1997-12-03	561-182-9077	30 Arapahoe Terrace	Orlando	FL	1	4	4	2021-01-29	4	2
10	Ketie	Thomas	1967-11-19	652-102-6790	68 Lawn Avenue	Atlanta	GA	5	10	10	2021-01-05	10	2

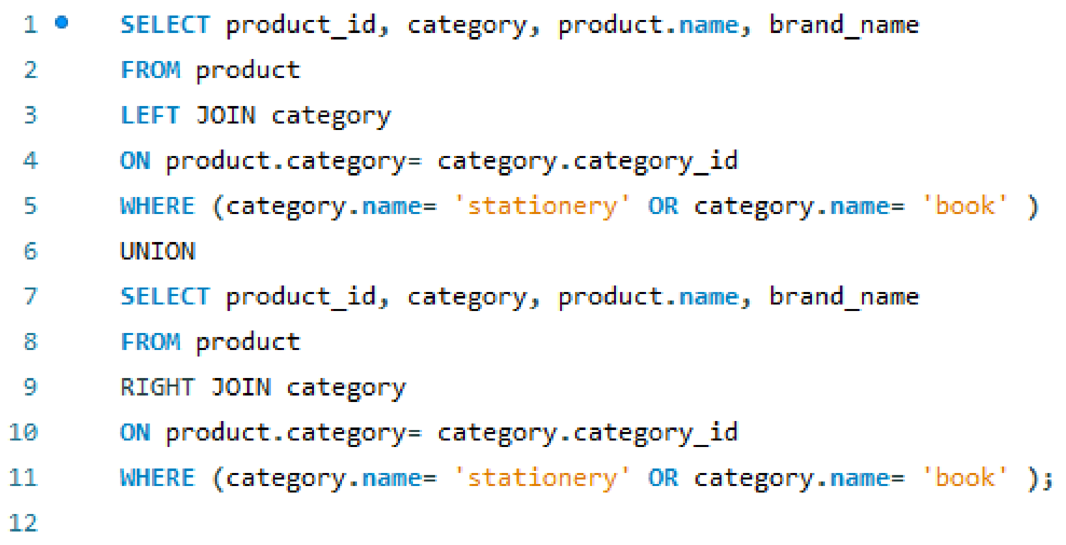
Figure 25: Left Join



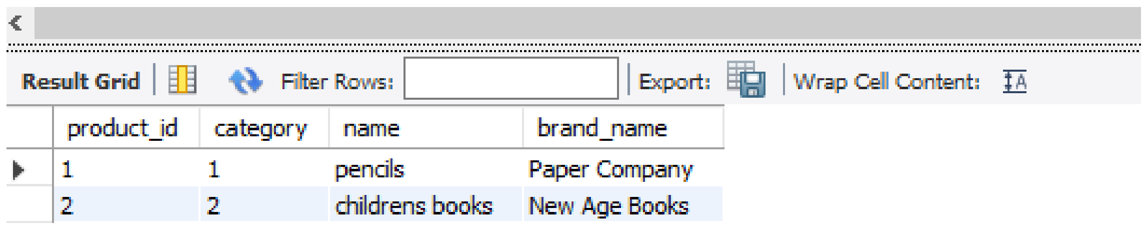
```sql
SELECT *
FROM employee
RIGHT JOIN employee_manager
ON employee.employee_id = employee_manager.manager_id;
```

employee_id	name	surname	birthday	phone	salary	employee_id	manager_id
NULL	NULL	NULL	NULL	NULL	NULL	101	NULL
NULL	NULL	NULL	NULL	NULL	NULL	102	NULL
NULL	NULL	NULL	NULL	NULL	NULL	105	NULL
101	Alex	Gordon	1986-06-17	559-181-3744	119241	598	101
101	Alex	Gordon	1986-06-17	559-181-3744	119241	688	101
102	Lisa	Woodson	1987-07-14	NULL	98926	771	102
102	Lisa	Woodson	1987-07-14	NULL	98926	900	102
105	Clara	Brown	1999-03-14	804-427-9456	110150	100	105
105	Clara	Brown	1999-03-14	804-427-9456	110150	303	105
105	Clara	Brown	1999-03-14	804-427-9456	110150	401	105

Figure 26: Right Join



```sql
SELECT product_id, category, product.name, brand_name
FROM product
LEFT JOIN category
ON product.category= category.category_id
WHERE (category.name= 'stationery' OR category.name= 'book' )
UNION
SELECT product_id, category, product.name, brand_name
FROM product
RIGHT JOIN category
ON product.category= category.category_id
WHERE (category.name= 'stationery' OR category.name= 'book' );
```

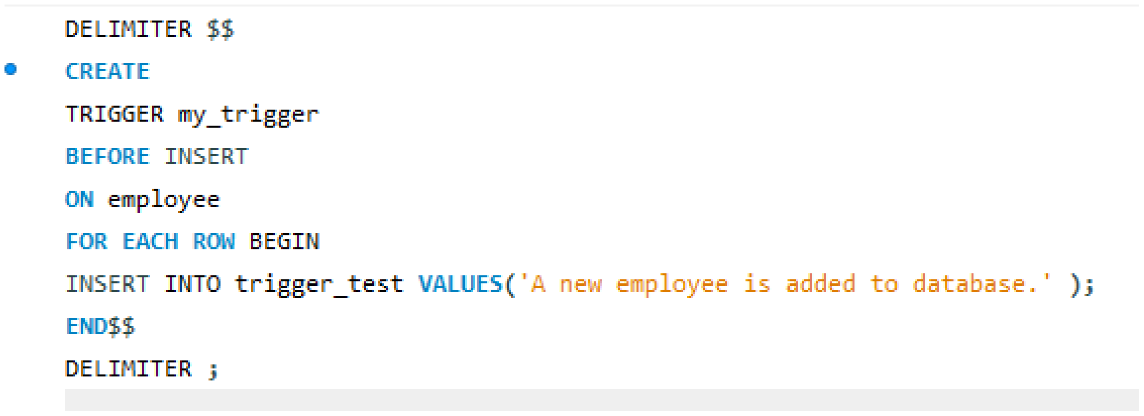


product_id	category	name	brand_name
1	1	pencils	Paper Company
2	2	childrens books	New Age Books

Figure 27: Outer Join

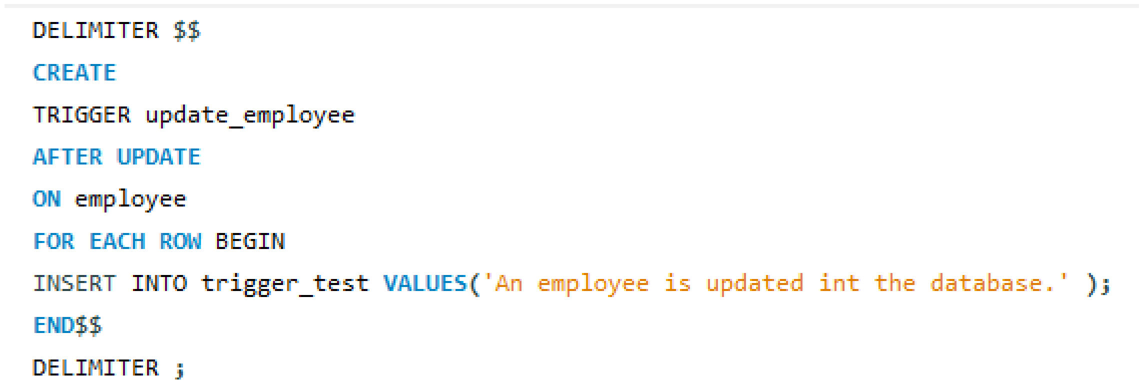
Since there is no full outer join in MySQL, I used UNION to make a left and right join.

7 Triggers



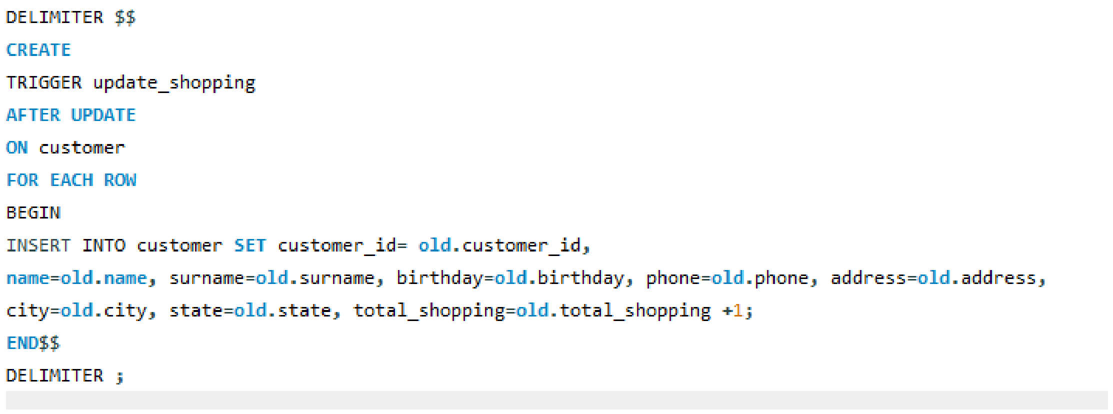
```
DELIMITER $$
CREATE
TRIGGER my_trigger
BEFORE INSERT
ON employee
FOR EACH ROW BEGIN
INSERT INTO trigger_test VALUES('A new employee is added to database.' );
END$$
DELIMITER ;
```

Figure 28: Trigger 1



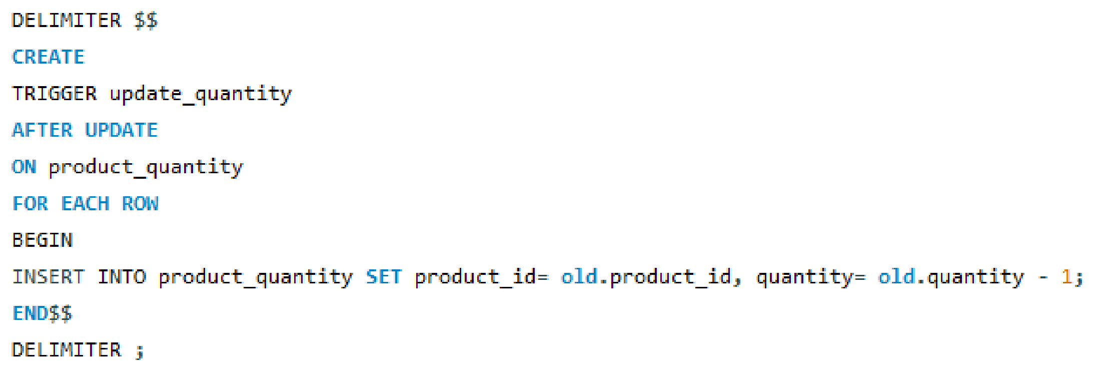
```
DELIMITER $$
CREATE
TRIGGER update_employee
AFTER UPDATE
ON employee
FOR EACH ROW BEGIN
INSERT INTO trigger_test VALUES('An employee is updated int the database.' );
END$$
DELIMITER ;
```

Figure 29: Trigger 2



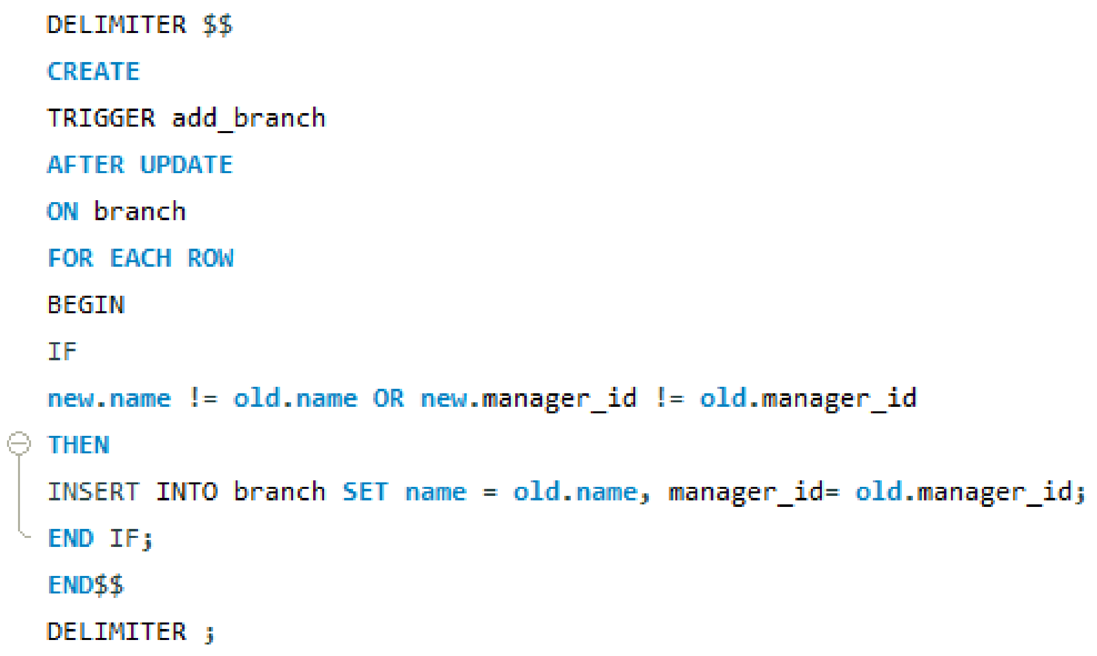
```
DELIMITER $$
CREATE
TRIGGER update_shopping
AFTER UPDATE
ON customer
FOR EACH ROW
BEGIN
INSERT INTO customer SET customer_id= old.customer_id,
name=old.name, surname=old.surname, birthday=old.birthday, phone=old.phone, address=old.address,
city=old.city, state=old.state, total_shopping=old.total_shopping +1;
END$$
DELIMITER ;
```

Figure 30: Trigger 3



```
DELIMITER $$
CREATE
TRIGGER update_quantity
AFTER UPDATE
ON product_quantity
FOR EACH ROW
BEGIN
INSERT INTO product_quantity SET product_id= old.product_id, quantity= old.quantity - 1;
END$$
DELIMITER ;
```

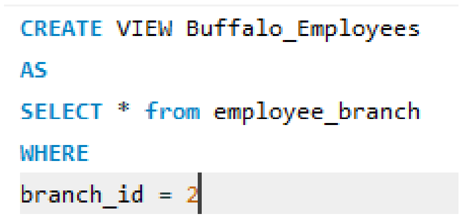
Figure 31: Trigger 4



```
DELIMITER $$
CREATE
TRIGGER add_branch
AFTER UPDATE
ON branch
FOR EACH ROW
BEGIN
IF
new.name != old.name OR new.manager_id != old.manager_id
THEN
INSERT INTO branch SET name = old.name, manager_id= old.manager_id;
END IF;
END$$
DELIMITER ;
```

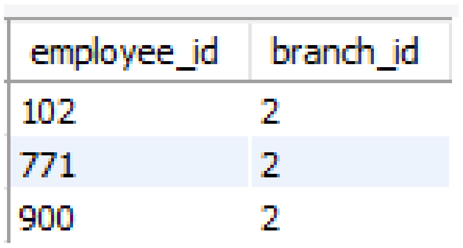
Figure 32: Trigger 5

8 Views



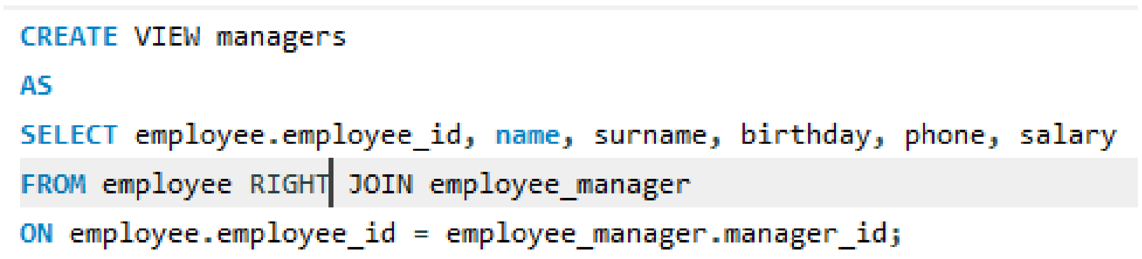
```
CREATE VIEW Buffalo_Employees
AS
SELECT * from employee_branch
WHERE
branch_id = 2
```

Figure 33: View 1



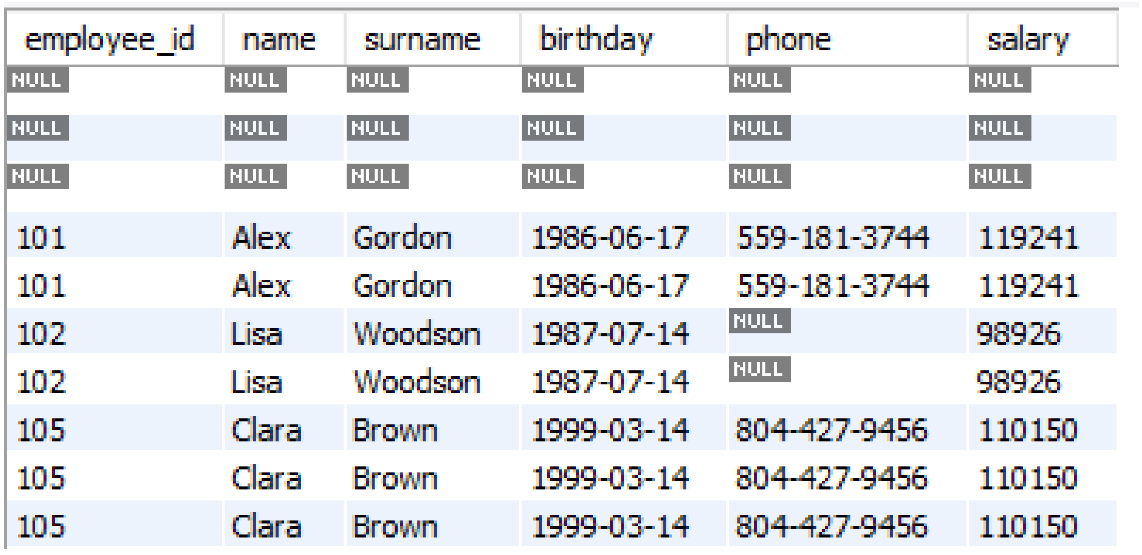
employee_id	branch_id
102	2
771	2
900	2

Figure 34: Employees of Buffalo Branch



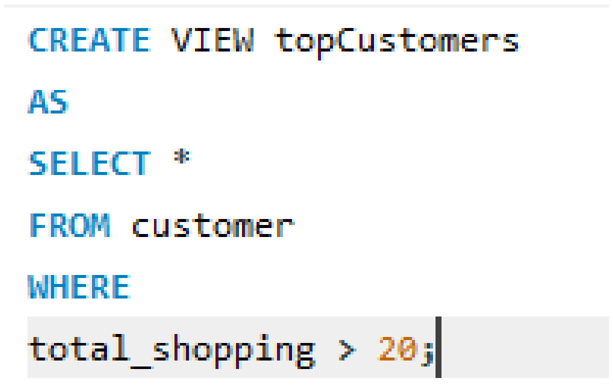
```
CREATE VIEW managers
AS
SELECT employee.employee_id, name, surname, birthday, phone, salary
FROM employee RIGHT JOIN employee_manager
ON employee.employee_id = employee_manager.manager_id;
```

Figure 35: View 2



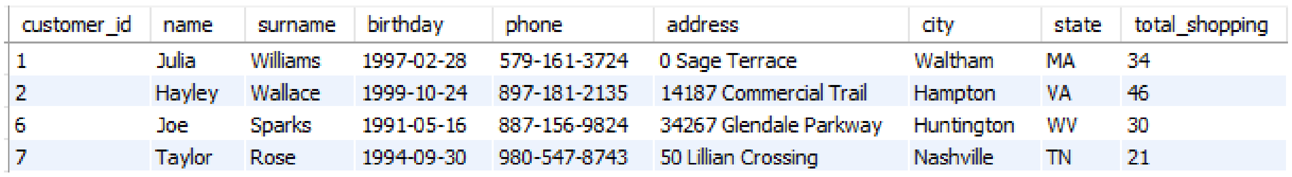
employee_id	name	surname	birthday	phone	salary
NULL	NULL	NULL	NULL	NULL	NULL
NULL	NULL	NULL	NULL	NULL	NULL
NULL	NULL	NULL	NULL	NULL	NULL
101	Alex	Gordon	1986-06-17	559-181-3744	119241
101	Alex	Gordon	1986-06-17	559-181-3744	119241
102	Lisa	Woodson	1987-07-14	NULL	98926
102	Lisa	Woodson	1987-07-14	NULL	98926
105	Clara	Brown	1999-03-14	804-427-9456	110150
105	Clara	Brown	1999-03-14	804-427-9456	110150
105	Clara	Brown	1999-03-14	804-427-9456	110150

Figure 36: Managers



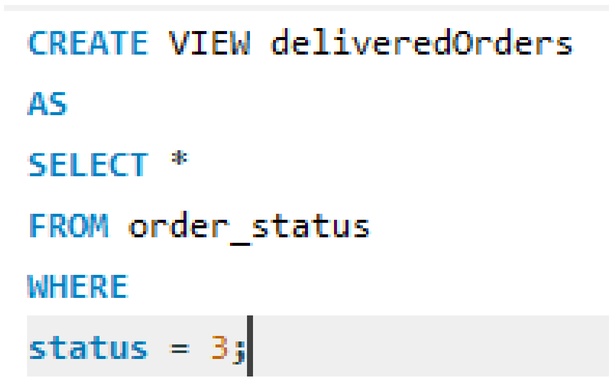
```
CREATE VIEW topCustomers
AS
SELECT *
FROM customer
WHERE
total_shopping > 20;
```

Figure 37: View 3



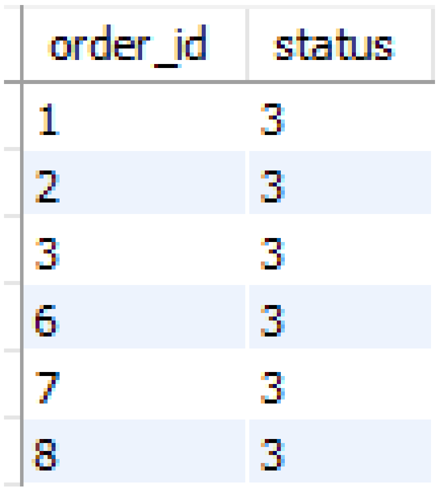
customer_id	name	surname	birthday	phone	address	city	state	total_shopping
1	Julia	Williams	1997-02-28	579-161-3724	0 Sage Terrace	Waltham	MA	34
2	Hayley	Wallace	1999-10-24	897-181-2135	14187 Commercial Trail	Hampton	VA	46
6	Joe	Sparks	1991-05-16	887-156-9824	34267 Glendale Parkway	Huntington	WV	30
7	Taylor	Rose	1994-09-30	980-547-8743	50 Lillian Crossing	Nashville	TN	21

Figure 38: Top customers - customers whose total shopping value is greater than 20



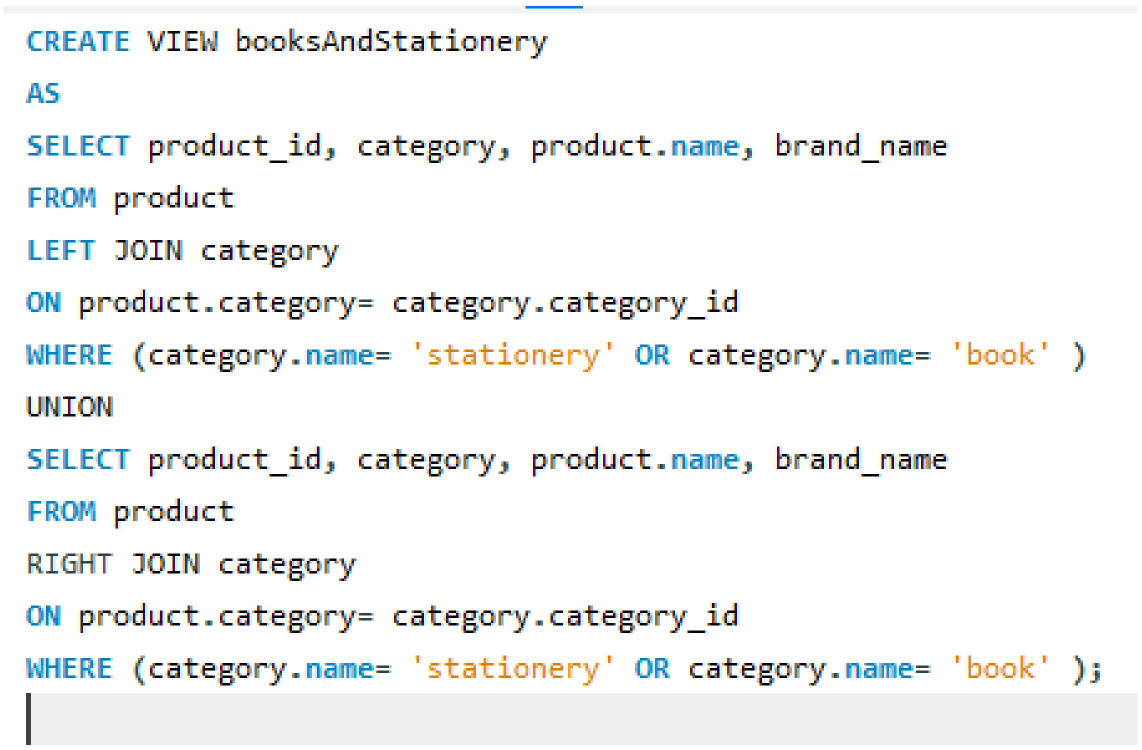
```sql
CREATE VIEW deliveredOrders
AS
SELECT *
FROM order_status
WHERE
status = 3;
```

Figure 39: View 4



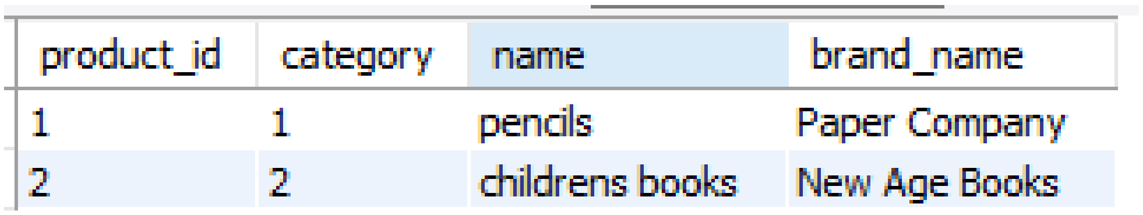
order_id	status
1	3
2	3
3	3
6	3
7	3
8	3

Figure 40: Delivered Orders



```
CREATE VIEW booksAndStationery
AS
SELECT product_id, category, product.name, brand_name
FROM product
LEFT JOIN category
ON product.category= category.category_id
WHERE (category.name= 'stationery' OR category.name= 'book' )
UNION
SELECT product_id, category, product.name, brand_name
FROM product
RIGHT JOIN category
ON product.category= category.category_id
WHERE (category.name= 'stationery' OR category.name= 'book' );
```

Figure 41: View 5



product_id	category	name	brand_name
1	1	pencils	Paper Company
2	2	childrens books	New Age Books

Figure 42: Products with books and stationery categories