

Create tables:

```
9 ✓ CREATE TABLE merchants (  
10     mid INT PRIMARY KEY,  
11     name VARCHAR(255),  
12     city VARCHAR(255),  
13     state VARCHAR(255)  
14 );  
15 ✓ CREATE TABLE products (  
16     pid INT PRIMARY KEY,  
17     name ENUM('Printer', 'Ethernet Adapter', 'Desktop', 'Hard Drive', 'Laptop', 'Router', 'Network Card', 'Super  
18     category ENUM('Peripheral', 'Networking', 'Computer'),  
19     description VARCHAR(255)  
20 );  
21  
22 ✓ CREATE TABLE sell (  
23     mid INT,  
24     pid INT,  
25     price DECIMAL(10, 2) CHECK (price BETWEEN 0 AND 100000),  
26     quantity_available INT CHECK (quantity_available BETWEEN 0 AND 1000),  
27     PRIMARY KEY (mid, pid),  
28     FOREIGN KEY (mid) REFERENCES merchants(mid),  
29     FOREIGN KEY (pid) REFERENCES products(pid)  
30 );  
31  
32 ✓ CREATE TABLE orders (  
33     oid INT PRIMARY KEY,  
34     shipping_method ENUM('UPS', 'FedEx', 'USPS'),  
35     shipping_cost DECIMAL(10, 2) CHECK (shipping_cost BETWEEN 0 AND 500)  
36 );  
37  
38 ✓ CREATE TABLE contain (  
39     oid INT,  
40     pid INT,  
41     PRIMARY KEY (oid, pid),  
42     FOREIGN KEY (oid) REFERENCES orders(oid),  
43     FOREIGN KEY (pid) REFERENCES products(pid)  
44 );
```

```
44 );
45
46 ✓ CREATE TABLE customers (
47     cid INT PRIMARY KEY,
48     fullname VARCHAR(255),
49     city VARCHAR(255),
50     state VARCHAR(255)
51 );
52 ✓ CREATE TABLE place (
53     cid INT,
54     oid INT,
55     order_date DATE,
56     PRIMARY KEY (cid, oid),
57     FOREIGN KEY (cid) REFERENCES customers(cid),
58     FOREIGN KEY (oid) REFERENCES orders(oid),
59     CHECK (order_date IS NULL OR STR_TO_DATE(order_date, '%Y-%m-%d') IS NOT NULL)
60 );
```

1.

```
1 ✓ select P.name as productName, M.name as SellerName
2   from products P
3  left join sell S on P.pid = S.pid
4  left join merchants M on S.mid = M.mid
5  where quantity_available = 0 or quantity_available is null;
```

2.

```
select P.name as productName, P.description as productDescription
from products P
left join sell S on P.pid = S.pid
where S.pid is null;
```

💡

Output Result 4 ×

2 rows

	productName	productDescription
1	Super Drive	External CD/DVD/RW
2	Super Drive	UInternal CD/DVD/RW

3.

Tx: Auto

Playground

1

✓

select count(distinct C.cid) as customerCount

2

from customers C

3

join place Pl on C.cid = Pl.cid

4

join contain Co on Pl.oid = Co.oid

5

join products P on Co.pid = P.pid

6

where P.category = 'SATA Drive'

7

and C.cid not in(

8

select distinct C.cid

9

from customers C

10

join place Pl on C.cid = Pl.cid

11

join contain Co on pl.oid = Co.oid

12

join products P on Co.pid = P.pid

13

where P.category = 'Router'

14

15

16

)

17

Output

customerCount:int

<

>

1 row

<

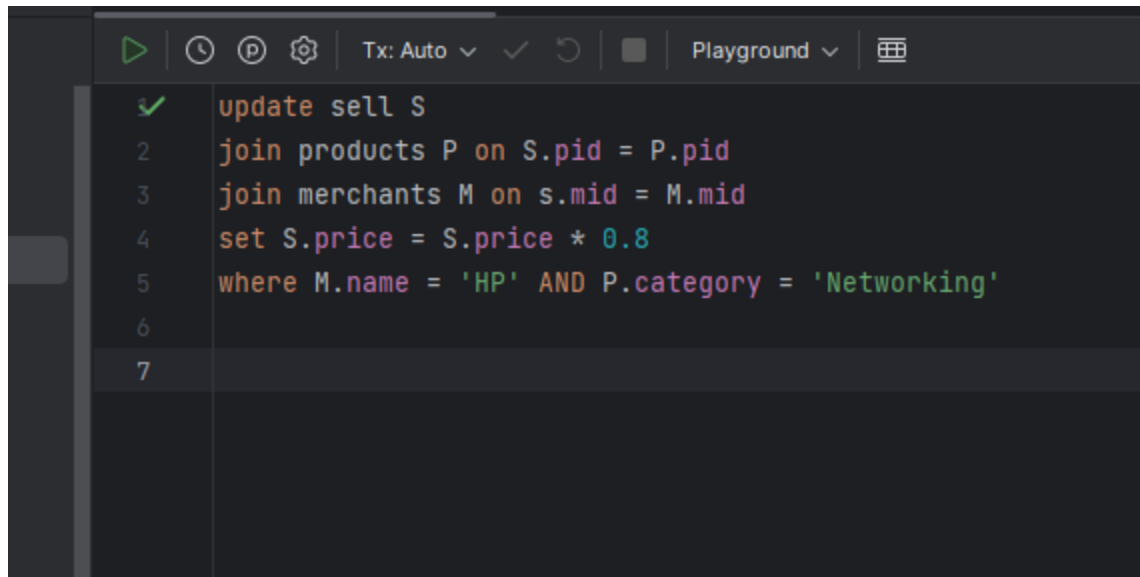
>

customerCount

1

0

4.



The image shows a SQL playground interface with a dark theme. At the top, there is a toolbar with icons for running, undo, redo, and settings, along with a dropdown menu set to 'Tx: Auto' and a 'Playground' dropdown. Below the toolbar, a list of line numbers (1-7) is on the left, and the corresponding SQL code is on the right. The code is as follows:

```
1 ✓ update sell S
2   join products P on S.pid = P.pid
3   join merchants M on s.mid = M.mid
4   set S.price = S.price * 0.8
5   where M.name = 'HP' AND P.category = 'Networking'
6
7
```

5.

1 ✓

2

3

4

5

6

7

8

9

10

```
select P.name as productsName,S.price as productPrice
from products P
join sell S on P.pid = S.pid
join contain C on P.pid = C.pid
join orders O on C.oid = O.oid
join place Pl on O.oid = Pl.oid
join customers Cu on Pl.cid = Cu.cid
join merchants M on S.mid = M.mid
where Cu.fullname = 'Uriel Whitney' and M.name = 'Acer'
```

Output

Result 10

✕

89 rows

⏮

⏪

⏩

⏭

🔄

🕒

🔍

📌

	productsName	productPrice
1	Super Drive	356.13
2	Network Card	130.43
3	Hard Drive	836.99
4	Printer	310.83
5	Printer	1345.37
6	Super Drive	356.13
7	Super Drive	1015.95
8	Network Card	405.40
9	Monitor	1103.47
10	Super Drive	1135.30
11	Router	1256.57

6.

▶

🕒

Ⓟ

⚙️

Tx: Auto

✓

↺

■

Playground

📄

hw333

console

1 ✓

2

3

4

5

6

7

8

```
SELECT YEAR(pl.order_date) AS year, m.name AS company, SUM(s.price) AS total_sales
FROM sell s
JOIN contain co ON s.pid = co.pid
JOIN orders o ON co.oid = o.oid
JOIN place pl ON o.oid = pl.oid
JOIN merchants m ON s.mid = m.mid
GROUP BY YEAR(pl.order_date), m.name
ORDER BY SUM(s.price) desc limit 1;
```

Output

Result 18

×

⏪

<

1 row

>

⏩

🔄

🕒

■

📌

	year	company	total_sales
1	2018	Lenovo	324291.59

8.

```
select AVG(o.shipping_cost) as avgShippingCost, o.shipping_method as shippingCompany
from orders o
group by o.shipping_method
order by AVG(o.shipping_cost) limit 1;
```

Output Result 44 ×

1 row ▾ | ↺ ⌚ | 📌

	avgShippingCost	shippingCompany
1	7.455761	USPS

9.

```

19
20 ✓ WITH CategorySales AS (
21     SELECT m.name AS company, p.category, SUM(s.price) AS total_sales
22     FROM sell s
23     JOIN contain co ON s.pid = co.pid
24     JOIN orders o ON co.oid = o.oid
25     JOIN place pl ON o.oid = pl.oid
26     JOIN merchants m ON s.mid = m.mid
27     JOIN products p ON s.pid = p.pid
28     GROUP BY m.name, p.category
29     ORDER BY m.name, total_sales DESC
30 )
31 SELECT company, category, total_sales
32 FROM (
33     SELECT company, category, total_sales,
34     ROW_NUMBER() OVER (PARTITION BY company ORDER BY total_sales DESC) AS rn
35     FROM CategorySales
36 ) AS ranked
37 WHERE rn = 1;
38
39

```

Output Result 24 x

5 rows

	company	category	total_sales
1	Acer	Peripheral	648729.57
2	Apple	Peripheral	613620.95
3	Dell	Peripheral	593504.38
4	HP	Peripheral	340861.72
5	Lenovo	Peripheral	608137.27

10.

59

40 ✓

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

```
WITH CustomerSpending AS (  
  SELECT m.name AS company, c.fullname AS customer, SUM(s.price) AS total_spent  
  FROM sell s  
  JOIN contain co ON s.pid = co.pid  
  JOIN orders o ON co.oid = o.oid  
  JOIN place pl ON o.oid = pl.oid  
  JOIN customers c ON pl.cid = c.cid  
  JOIN merchants m ON s.mid = m.mid  
  GROUP BY m.name, c.fullname  
)  
SELECT company, customer, total_spent  
FROM (  
  SELECT company, customer, total_spent,  
         ROW_NUMBER() OVER (PARTITION BY company ORDER BY total_spent DESC) AS rn_highest,  
         ROW_NUMBER() OVER (PARTITION BY company ORDER BY total_spent ASC) AS rn_lowest  
  FROM CustomerSpending  
) AS ranked  
WHERE rn_highest = 1 OR rn_lowest = 1;
```

ranked

Output

Result 26

10 rows

company

customer

total_spent

1	Acer	Inez Long	31901.02
2	Acer	Dean Heath	75230.29
3	Apple	Inez Long	32251.10
4	Apple	Clementine Travis	84551.11
5	Dell	Inez Long	31135.74
6	Dell	Clementine Travis	85611.55
7	HP	Inez Long	21420.58
8	HP	Clementine Travis	55063.59
9	Lenovo	Inez Long	33948.91
10	Lenovo	Haviva Stewart	83030.26

YuningHW31