

- 1

10

[illegible]

```

SELECT
  * FROM products LIMIT 10;
CREATE TABLE products (
  pid INT PRIMARY KEY,
  name VARCHAR(255),
  category VARCHAR(100),
  description TEXT
);

```

100% 20:20

Result Grid Filter Rows: Search Edit: Export/Import: Fetch

pid	name	category	description
1	Hard Drive	Peripheral	640GB USB 2.0 Portable
2	Monitor	Peripheral	LED 22-inch Backlit
3	Printer	Peripheral	All-in-one
4	Super Drive	Peripheral	8X DVD-ROM 32X CD-ROM SATA Internal Blu-...
5	Hard Drive	Peripheral	2TB Internal SATA
6	Laptop	Computer	Intel Core i5-2410M 2.3GHz / 4GB RAM / 640G...
7	Super Drive	Peripheral	DVD+R 8X DVD+RW 22X DVD-R 16X DVD-RO...
8	Router	Networking	Gigabit Router with USB for Windows and Linux
9	Monitor	Peripheral	23-inch HD Widescreen
10	Network Card	Networking	Wireless a/b/g/n
NULL	NULL	NULL	NULL

products 36 Apply Revert

Result Grid Form Editor Field Types

100%

- `SELECT *`
 `FROM orders LIMIT 10;`
- `CREATE TABLE orders (`
 `oid INT PRIMARY KEY,`
 `shipping_method VARCHAR(50),`
 `shipping_cost DECIMAL(10, 2)`
 `);`

[illegible]

①

[illegible]

```

* SELECT
  * FROM customers LIMIT 10;
* ⊖ CREATE TABLE customers (
    cid INT PRIMARY KEY,
    fullname VARCHAR(255),
    city VARCHAR(100),
    state VARCHAR(50)
  );

```

[illegible]


```
-- 1. List names and sellers of products that are no longer available (quantity = 0)
• SELECT p.name AS product_name, m.name AS merchant_name
  FROM products p
  JOIN sell s ON p.pid = s.pid
  JOIN merchants m ON s.mid = m.mid
 WHERE s.quantity_available = 0;
```

Explanation: This query retrieves the names of products and their sellers where the product's quantity available is 0. It joins the products, sell, and merchants tables to get the necessary details.

```
-- 2. List names and descriptions of products that are not sold
SELECT p.name, p.description
FROM products p
LEFT JOIN sell s ON p.pid = s.pid
WHERE s.pid IS NULL;
```

Explanation: This query fetches product names and descriptions for products that have not been sold. It uses a LEFT JOIN between products and sell, ensuring it includes all products and only those not linked to a sale (WHERE s.pid IS NULL).


```

-- 3. How many customers bought SATA drives but not any routers?
SELECT COUNT(DISTINCT c.cid) AS customer_count
FROM customers c
JOIN place pl ON c.cid = pl.cid
JOIN contain ct ON pl.oid = ct.oid
JOIN products p ON ct.pid = p.pid
WHERE p.name LIKE '%SATA%'
AND c.cid NOT IN (
    SELECT c2.cid
    FROM customers c2
    JOIN place pl2 ON c2.cid = pl2.cid
    JOIN contain ct2 ON pl2.oid = ct2.oid
    JOIN products p2 ON ct2.pid = p2.pid
    WHERE p2.name LIKE '%Router%'
);

```

Explanation: This query counts the distinct customers who bought SATA drives but not routers. The NOT IN clause excludes customers who purchased routers, ensuring only those who bought SATA drives are counted.

```

-- 4. Apply a 20% sale on HP's Networking products
UPDATE sell s
JOIN products p ON s.pid = p.pid
JOIN merchants m ON s.mid = m.mid
SET s.price = s.price * 0.80
WHERE m.name = 'HP' AND p.category = 'Networking';

```

Explanation: This query applies a 20% discount on all networking products sold by HP. It uses a JOIN to connect the sell, products, and merchants tables, and updates the price by reducing it by 20%.

```

-- 5. Retrieve what Uriel Whitney ordered from Acer (product names and prices)
• SELECT p.name AS product_name, s.price
  FROM customers c
  JOIN place pl ON c.cid = pl.cid
  JOIN contain ct ON pl.oid = ct.oid
  JOIN products p ON ct.pid = p.pid
  JOIN sell s ON p.pid = s.pid
  JOIN merchants m ON s.mid = m.mid
  WHERE c.fullname = 'Uriel Whitney' AND m.name = 'Acer';

```

Explanation: This query fetches the product names and prices for orders placed by Uriel Whitney from Acer. It joins several tables to retrieve the required information based on the customer's name and the merchant's name.

```

-- 6. List the annual total sales for each company
SELECT m.name AS merchant_name, YEAR(pl.order_date) AS year, SUM(s.price * ct.quantity) AS total_sales
FROM merchants m
JOIN sell s ON m.mid = s.mid
JOIN contain ct ON s.pid = ct.pid
JOIN place pl ON ct.oid = pl.oid
GROUP BY m.name, YEAR(pl.order_date)
ORDER BY m.name, year;

```

Explanation: This query lists the annual total sales for each merchant by joining the merchants, sell, contain, and place tables, summing up the product prices and quantities sold each year.

```

-- 7. Which company had the highest annual revenue and in what year?
• SELECT merchant_name, year, total_sales
  FROM (
    SELECT m.name AS merchant_name, YEAR(pl.order_date) AS year, SUM(s.price * ct.quantity) AS total_sales
    FROM merchants m
    JOIN sell s ON m.mid = s.mid
    JOIN contain ct ON s.pid = ct.pid
    JOIN place pl ON ct.oid = pl.oid
    GROUP BY m.name, YEAR(pl.order_date)
  ) AS annual_sales
  ORDER BY total_sales DESC
  LIMIT 1;

```

Explanation: This query calculates the highest annual revenue for any company by first calculating the yearly sales totals for each merchant, then selecting the one with the highest total using ORDER BY total_sales DESC LIMIT 1

```

-- 8. On average, what was the cheapest shipping method used ever?
• SELECT shipping_method, AVG(shipping_cost) AS avg_cost
  FROM orders
  GROUP BY shipping_method
  ORDER BY avg_cost ASC
  LIMIT 1;

```

Explanation: This query finds the cheapest shipping method by calculating the average shipping cost for each method and selecting the one with the lowest average cost.

```

-- 9. What is the best sold ($) category for each company?
SELECT m.name AS merchant_name, p.category, SUM(s.price * ct.quantity) AS total_sales
  FROM merchants m
  JOIN sell s ON m.mid = s.mid
  JOIN products p ON s.pid = p.pid
  JOIN contain ct ON s.pid = ct.pid
  JOIN place pl ON ct.oid = pl.oid
  GROUP BY m.name, p.category
  ORDER BY merchant_name, total_sales DESC;

```

Explanation: This query identifies the highest revenue-generating product category for each merchant by summing the total sales (price * quantity) for each category and ordering them by merchant and total sales.

```

-- 10. For each company, find out which customers have spent the most and the least amounts
WITH CustomerSpend AS (
    SELECT m.name AS merchant_name, c.fullname, SUM(s.price * ct.quantity) AS total_spent
    FROM merchants m
    JOIN sell s ON m.mid = s.mid
    JOIN contain ct ON s.pid = ct.pid
    JOIN place pl ON ct.oid = pl.oid
    JOIN customers c ON pl.cid = c.cid
    GROUP BY m.name, c.fullname
)
SELECT merchant_name, fullname, total_spent
FROM CustomerSpend
WHERE (merchant_name, total_spent) IN (
    SELECT merchant_name, MAX(total_spent) FROM CustomerSpend GROUP BY merchant_name
    UNION ALL
    SELECT merchant_name, MIN(total_spent) FROM CustomerSpend GROUP BY merchant_name
);

```

Explanation: This query calculates which customers have spent the most and the least for each merchant by first calculating the total amount spent per customer (CustomerSpend) and then selecting the max and min totals for each merchant.