1. **Write a SELECT statement that joins the Categories table to the Products table and returns these columns: category\_name, product\_name, list\_price.**

**Sort the result set by category\_name and then by product\_name in ascending sequence.**

SELECT

c.category\_name,

p.product\_name,

p.list\_price

FROM

Categories c

JOIN

Products p

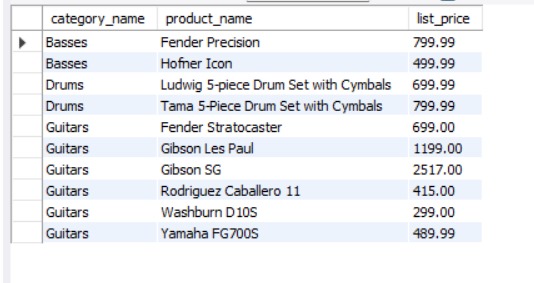
ON

c.category\_id = p.category\_id

ORDER BY

c.category\_name ASC,

p.product\_name ASC;



1. **Write a SELECT statement that joins the Customers table to the Addresses table and returns these columns: first\_name, last\_name, line1, city, state, zip\_code.**

**Return one row for each address for the customer with an email address of** [**allan.sherwood@yahoo.com**](mailto:allan.sherwood@yahoo.com)**.**

SELECT

c.first\_name,

c.last\_name,

a.line1,

a.city,

a.state,

a.zip\_code

FROM

Customers c

JOIN

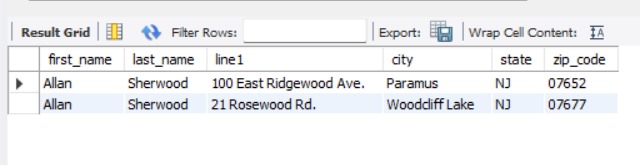
Addresses a

ON

c.customer\_id = a.customer\_id

WHERE

c.email\_address = 'allan.sherwood@yahoo.com';



1. **Write a SELECT statement that joins the Customers table to the Addresses table and returns these columns: first\_name, last\_name, line1, city, state, zip\_code.**

**Return one row for each customer, but only return addresses that are the shipping address for a customer.**

SELECT

c.first\_name,

c.last\_name,

a.line1,

a.city,

a.state,

a.zip\_code

FROM

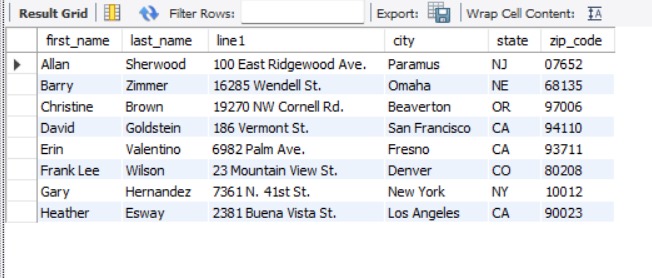
Customers c

JOIN

Addresses a

ON

c.shipping\_address\_id = a.address\_id;



1. **Write a SELECT statement that joins the Customers, Orders, Order\_Items, and Products tables. Thisstatement should return these columns: last\_name, first\_name, order\_date, product\_name, item\_price, discount\_amount, and quantity.**

**Use aliases for the tables. Sort the final result set by last\_name, order\_date, and product\_name.**

|  |  |
| --- | --- |
| SELECT  c.last\_name,  c.first\_name,  o.order\_date,  p.product\_name,  oi.item\_price,  oi.discount\_amount,  oi.quantity  FROM  Customers AS c  JOIN  Orders AS o  ON  c.customer\_id = o.customer\_id  JOIN  Order\_Items AS oi  ON o.order\_id = oi.order\_id  JOIN  Products AS p  ON  oi.product\_id = p.product\_id  ORDER BY  c.last\_name,  o.order\_date,  p.product\_name; |  |

1. **Write a SELECT statement that returns the product\_name and list\_price columns from the Products table. Return one row for each product that has the same list price as another product.**

**Hint: Use a self-join to check that the product\_id columns aren’t equal but the list\_price columns are equal. Sort the result set by product\_name**

SELECT

p1.product\_name,

p1.list\_price

FROM

Products AS p1

JOIN

Products AS p2

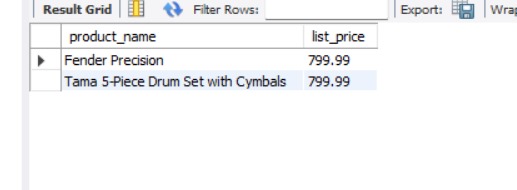
ON

p1.product\_id <> p2.product\_id

AND p1.list\_price = p2.list\_price

ORDER BY

p1.product\_name;



1. **Write a SELECT statement that returns these two columns:**

**category\_name The category\_name column from the Categories table product\_id The product\_id column from the Products table**

**Return one row for each category that has never been used.**

**Hint: Use an outer join and only return rows where the product\_id column contains a null value.**

SELECT

c.category\_name,

p.product\_id

FROM

Categories AS c

LEFT JOIN

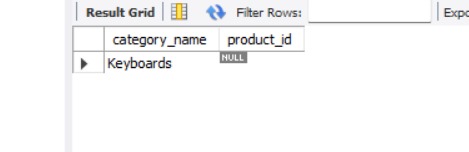
Products AS p

ON

c.category\_id = p.category\_id

WHERE

p.product\_id IS NULL;



1. **Use the UNION operator to generate a result set consisting of three columns from the Orders table: ship\_status A calculated column that contains a value of SHIPPED or NOT SHIPPED order\_id The order\_id column order\_date The order\_date column If the order has a value in the ship\_date column, the ship\_status column should contain a value of SHIPPED. Otherwise, it should contain a value of NOT SHIPPED. Sort the final result set by order\_date.**

SELECT

'SHIPPED' AS ship\_status,

order\_id,

order\_date

FROM

Orders

WHERE

ship\_date IS NOT NULL

UNION

SELECT

'NOT SHIPPED' AS ship\_status,

order\_id,

order\_date

FROM

Orders

WHERE

ship\_date IS NULL

ORDER BY

order\_date;

