DESCRIPTION

Table: Prices

+---------------+---------+

| Column Name | Type |

+---------------+---------+

| product_id | int |

| start_date | date |

| end_date | date |

| price | int |

+---------------+---------+

(product_id, start_date, end_date) is the primary key (combination of columns with unique values) for this table.

Each row of this table indicates the price of the product_id in the period from start_date to end_date.

For each product_id there will be no two overlapping periods. That means there will be no two intersecting periods for the same product_id.

Table: UnitsSold

+---------------+---------+

| Column Name | Type |

+---------------+---------+

| product_id | int |

| purchase_date | date |

| units | int |

+---------------+---------+

This table may contain duplicate rows.

Each row of this table indicates the date, units, and product_id of each product sold.

Write a solution to find the average selling price for each product. average_price should be **rounded to 2 decimal places**. If a product does not have any sold units, its average selling price is assumed to be 0.

Return the result table in **any order**.

The result format is in the following example.

Example 1:

Input:

Prices table:

```
+------------+------------+------------+--------+
| product_id | start_date | end_date   | price  |
+------------+------------+------------+--------+
| 1          | 2019-02-17 | 2019-02-28 | 5      |
| 1          | 2019-03-01 | 2019-03-22 | 20     |
| 2          | 2019-02-01 | 2019-02-20 | 15     |
| 2          | 2019-02-21 | 2019-03-31 | 30     |
+------------+------------+------------+--------+
```

UnitsSold table:

```
+------------+---------------+-------+
| product_id | purchase_date | units |
+------------+---------------+-------+
| 1          | 2019-02-25    | 100   |
| 1          | 2019-03-01    | 15    |
| 2          | 2019-02-10    | 200   |
| 2          | 2019-03-22    | 30    |
+------------+---------------+-------+
```

Output:

```
+------------+---------------+
| product_id | average_price |
+------------+---------------+
| 1          | 6.96          |
| 2          | 16.96         |
+------------+---------------+
```

Explanation:

Average selling price = Total Price of Product / Number of products sold.

Average selling price for product 1 = ((100 * 5) + (15 * 20)) / 115 = 6.96

Average selling price for product 2 = ((200 * 15) + (30 * 30)) / 230 = 16.96

SOLUTION

MySQL:

- Find average_price using SUM(), replace null with zero using IFNULL() and ROUND the result to 2 decimals using ROUND()
- Join tables using LEFT JOIN and GROUP BY with product_id

```
SELECT p.product_id, ROUND(IFNULL(SUM(p.price*u.units)/SUM(u.units), 0), 2) average_price
FROM Prices p
LEFT JOIN UnitsSold u
ON p.product_id = u.product_id AND u.purchase_date BETWEEN p.start_date AND p.end_date
GROUP BY p.product_id;
```

PostgreSQL:

- Find average_price using SUM(), replace null with zero using COALESCE() and ROUND the result to 2 decimals using ROUND()
- Join tables using LEFT JOIN and GROUP BY with product_id

```
SELECT p.product_id, COALESCE(ROUND(SUM(p.price*u.units*1.0)/SUM(u.units),2),0) average_price
FROM Prices p
LEFT JOIN UnitsSold u
ON u.product_id = p.product_id AND u.purchase_date BETWEEN p.start_date AND p.end_date
GROUP BY 1;
```