QUESTION

Product Sales by Month

Medium

10 Points

Given a table containing data for monthly sales, write a query to find the total amount of each product sold for each month, with each product as its own column in the output table.

Output Schema:

Column	Type
month	DATE
product_1	INT
product_2	INT
product_3	INT
product_4	INT

TABLE SCHEMA

```sql
CREATE TABLE monthly_sales (
month DATE,
product_id INTEGER,
amount_sold INTEGER
);

INSERT INTO monthly_sales (month, product_id, amount_sold) VALUES
('2021-01-01', 1, 100),
('2021-01-01', 2, 300),
('2021-02-01', 1, 150),
('2021-02-01', 1, 50),
('2021-02-01', 2, 250),
('2021-03-01', 1, 120),
('2021-03-01', 4, 250),
('2021-04-01', 2, -30),
('2021-04-01', 3, 200),
('2021-05-01', 3, 175),
('2021-06-01', 1, 0),
('2021-06-01', 2, 100);
```

SOLUTION

OUTPUT

month	product_1	product_2	product_3	product_4
2021-01-01	100	300	0	0
2021-02-01	200	250	0	0
2021-03-01	120	0	0	250
2021-04-01	0	-30	200	0
2021-05-01	0	0	175	0
2021-06-01	0	100	0	0

My Thought Process:

When I saw the structure, I immediately thought of a pivot-style transformation. Since SQL doesn't have a built-in pivot in all engines, I used CASE WHEN inside SUM() for each product. I grouped by month to get totals and carefully checked for duplicate product entries (like two entries for product 1 in one month). The tricky part was making sure all products were represented even if their sales were zero in certain months.

Business Impact:

This kind of query is perfect for generating monthly performance dashboards. Imagine a retail business using this to compare how different products perform month over month identifying trends, seasonal spikes, or inventory mismatches. It can help marketing teams decide where to focus promotions or help procurement optimize stock levels based on historical demand.