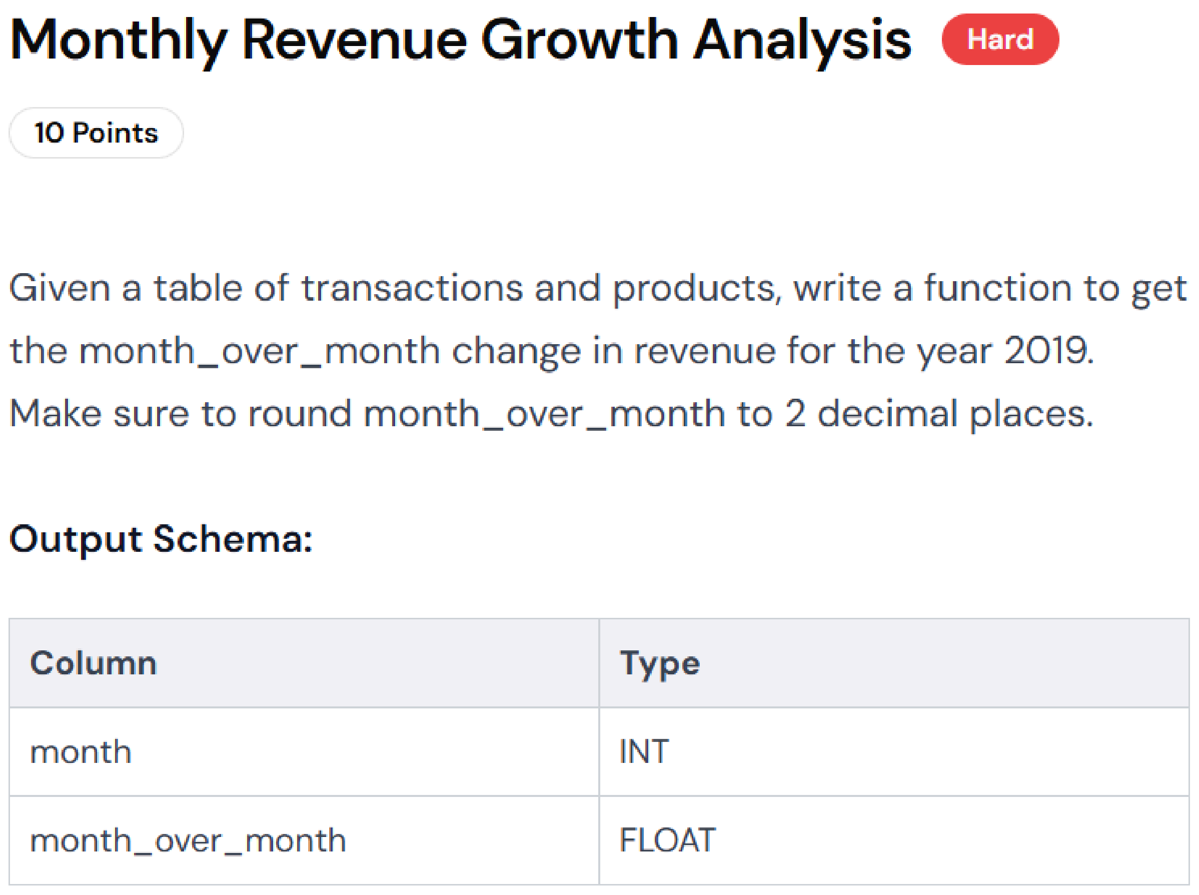
QUESTION



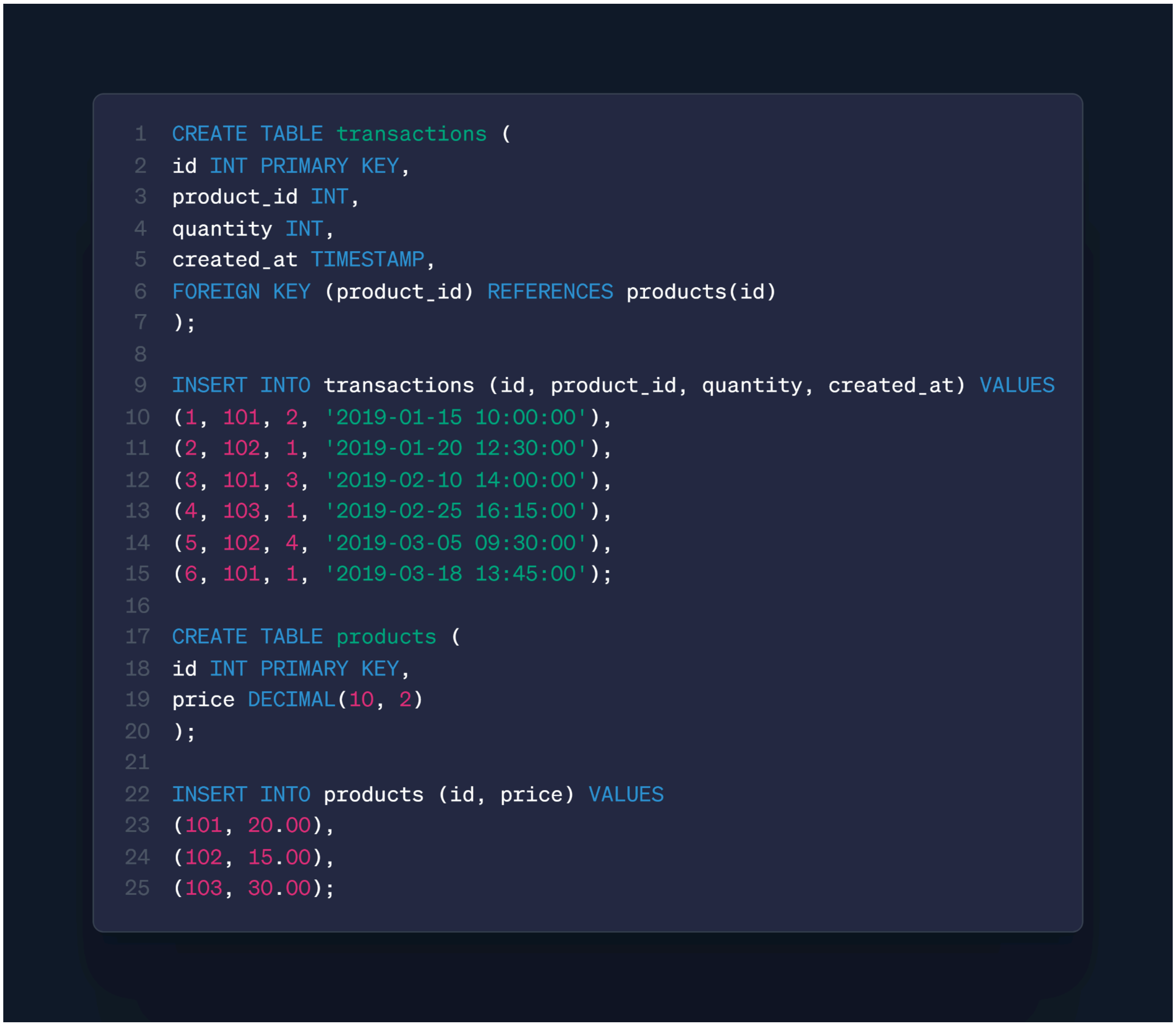
Monthly Revenue Growth Analysis

Given a table of transactions and products, write a function to get the month_over_month change in revenue for the year 2019. Make sure to round month_over_month to 2 decimal places.

Output Schema:

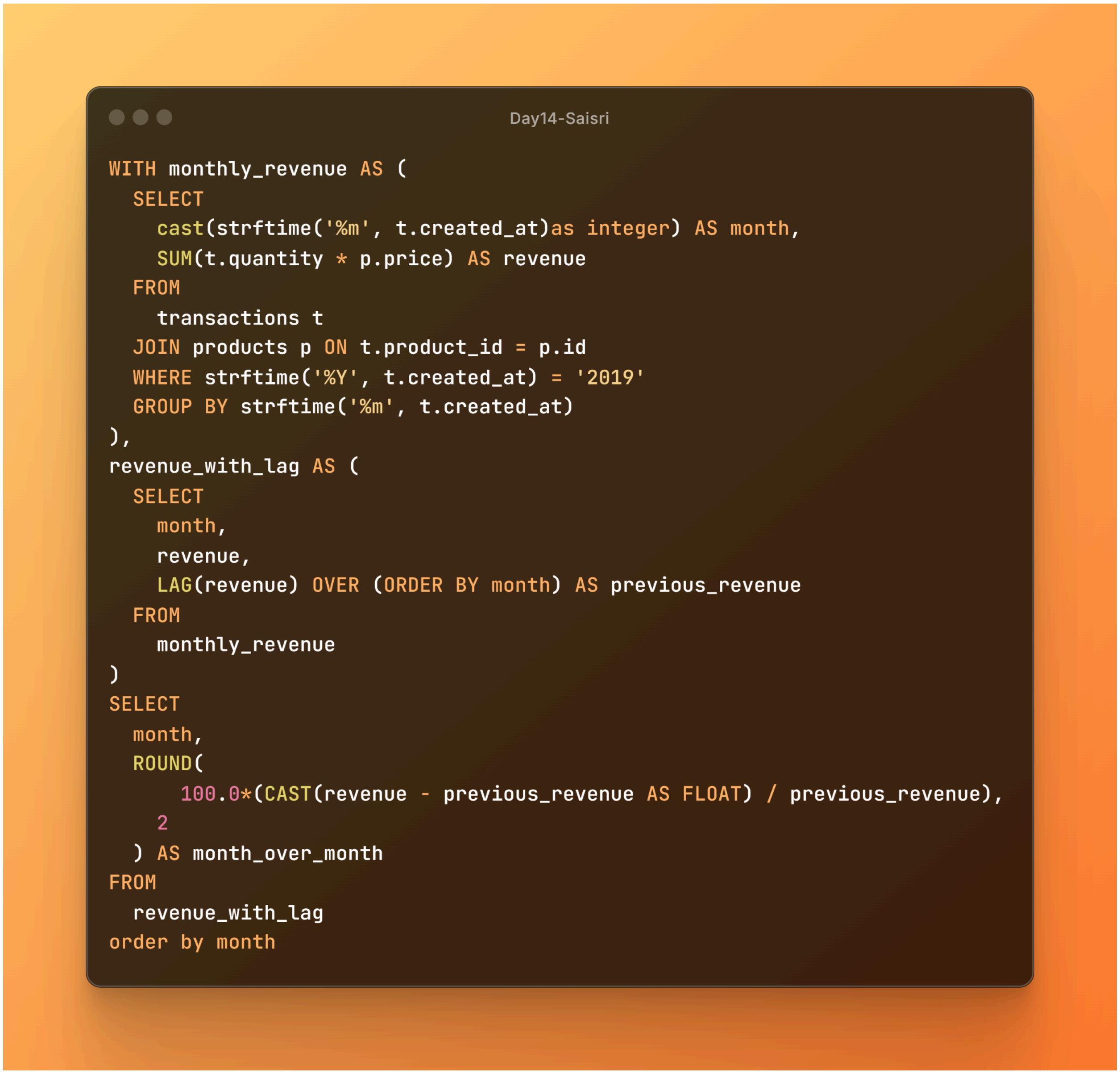
Column	Type
month	INT
month_over_month	FLOAT

TABLE SCHEMA



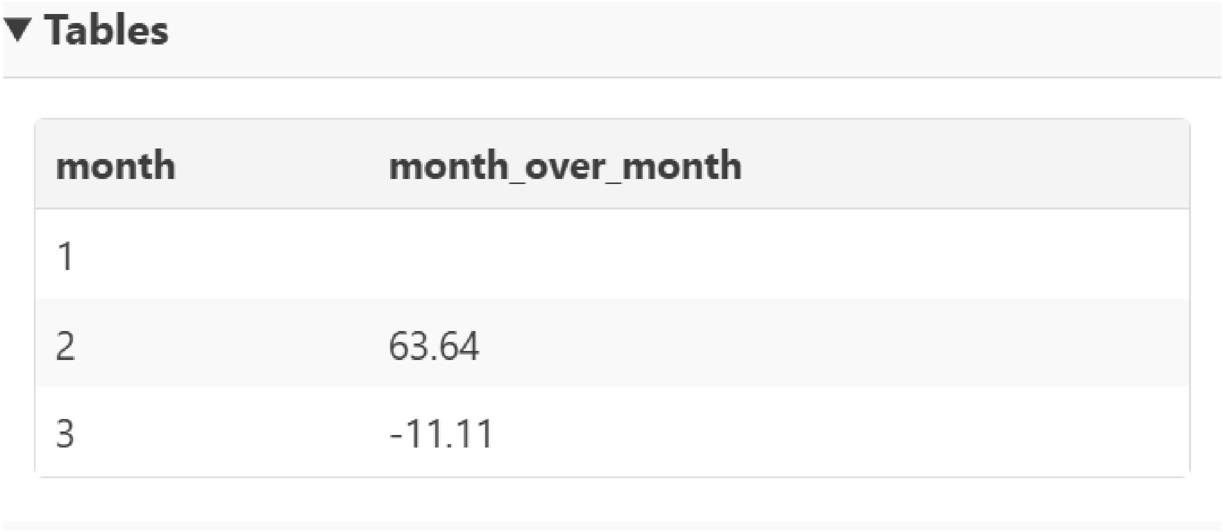
```sql
CREATE TABLE transactions (
id INT PRIMARY KEY,
product_id INT,
quantity INT,
created_at TIMESTAMP,
FOREIGN KEY (product_id) REFERENCES products(id)
);

INSERT INTO transactions (id, product_id, quantity, created_at) VALUES
(1, 101, 2, '2019-01-15 10:00:00'),
(2, 102, 1, '2019-01-20 12:30:00'),
(3, 101, 3, '2019-02-10 14:00:00'),
(4, 103, 1, '2019-02-25 16:15:00'),
(5, 102, 4, '2019-03-05 09:30:00'),
(6, 101, 1, '2019-03-18 13:45:00');

CREATE TABLE products (
id INT PRIMARY KEY,
price DECIMAL(10, 2)
);

INSERT INTO products (id, price) VALUES
(101, 20.00),
(102, 15.00),
(103, 30.00);
```

SOLUTION



```
WITH monthly_revenue AS (
  SELECT
    cast(strftime('%m', t.created_at)as integer) AS month,
    SUM(t.quantity * p.price) AS revenue
  FROM
    transactions t
  JOIN products p ON t.product_id = p.id
  WHERE strftime('%Y', t.created_at) = '2019'
  GROUP BY strftime('%m', t.created_at)
),
revenue_with_lag AS (
  SELECT
    month,
    revenue,
    LAG(revenue) OVER (ORDER BY month) AS previous_revenue
  FROM
    monthly_revenue
)
SELECT
  month,
  ROUND(
      100.0*(CAST(revenue - previous_revenue AS FLOAT) / previous_revenue),
    2
  ) AS month_over_month
FROM
  revenue_with_lag
order by month
```

OUTPUT



month	month_over_month
1	
2	63.64
3	-11.11

My Thought Process:

I joined the transactions and products tables to get the price for each item, then grouped the data by month to calculate total revenue. After that, I used the LAG() function to get the previous month's revenue and calculated the percentage change.

Business Impact:

For example, a sudden drop might signal seasonality or a product issue helping teams act fast to investigate or adapt strategies like promotions, stock adjustments, or feature rollbacks.