CREATE TABLE SCRIPT:

```sql
CREATE TABLE transactions (
    buyer_id INT,
    store_id VARCHAR(5),
    item_id VARCHAR(10),
    gross_transaction_value INT,
    purchase_time DATETIME,
    refund_item DATETIME DEFAULT NULL
);

INSERT INTO transactions VALUES
(4, 'a', 'D2', 58, '2019-09-04 04:48:35', NULL),
(3, 'b', 'b2', 475, '2020-04-11 19:24:47', NULL),
(12, 'b', 'b2', 475, '2020-04-11 19:24:47', '2020-04-16 22:28:00'),
(3, 'f', 'f9', 33, '2020-09-01 23:59:46', '2020-09-03 05:00:00'),
(7, 'd', 'd3', 250, '2020-10-23 21:35:09', NULL),
(9, 'e', 'e7', 24, '2020-04-02 15:38:25', NULL),
(5, 'g', 'g6', 61, '2020-10-15 12:13:02', '2020-10-18 02:53:00');


CREATE TABLE items (
    item_id VARCHAR(10),
    item_name VARCHAR(50)
);

INSERT INTO items VALUES
('D2', 'jeans'),
('b2', 'shirt'),
('f9', 'denim pants'),
('d3', 't-shirt'),
('e7', 'crocs'),
('g6', 'wall paint');
```

100% 68:20 2 errors found

Action Output

	Time	Action	Response	Duration / Fetch Time
6	19:10:19	INSERT INTO items VALUES ('D2', 'jeans'), ('b2', 'shirt'), ('f9', 'denim pants'), ('d3', 't-shirt'), ('e7', 'crocs'), ('g6', 'wall...	6 row(s) affected Records: 6 Duplicates: 0 Warnings...	0.0028 sec

Create table

Query 1 : Count of purchases per month (excluding refunded purchases) Any transaction with a refund_item timestamp means the purchase is considered refunded, so it is excluded from counting valid purchases.

```sql
SELECT
    DATE_FORMAT(purchase_time, '%Y-%m') AS purchase_month,
    COUNT(*) AS total_purchases
FROM transactions
WHERE refund_item IS NULL
GROUP BY DATE_FORMAT(purchase_time, '%Y-%m')
ORDER BY purchase_month;
```

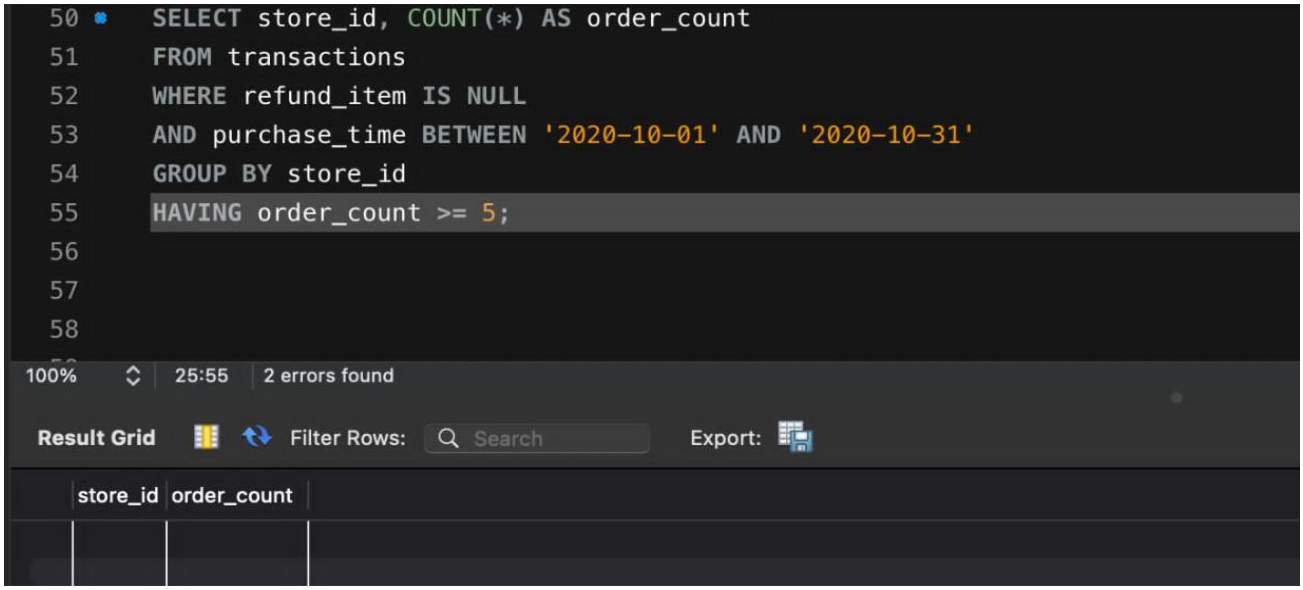
100% 25:47 2 errors found

Result Grid Filter Rows: Search Export:

purchase_mon...	total_purchas...
2019-09	1
2020-04	2
2020-10	1

Explanation: Groups non-refunded purchases by month using DATE_FORMAT.

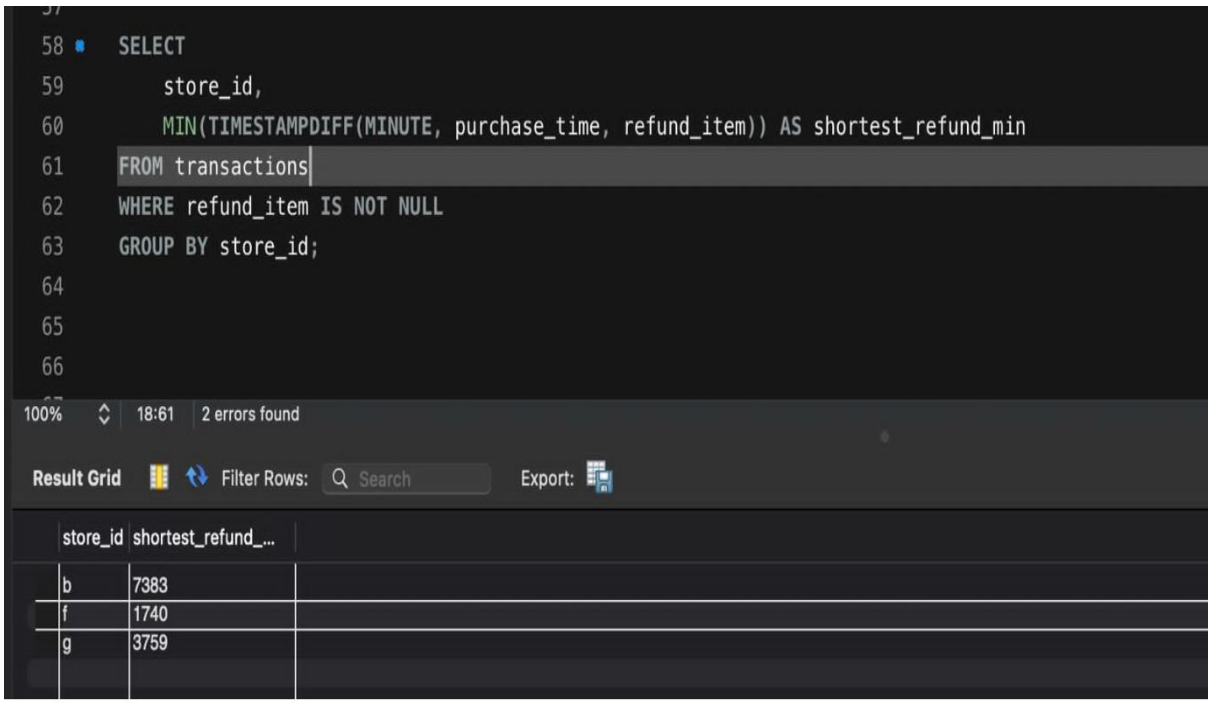
Query 2 : Stores with ≥ 5 transactions in October 2020 Only purchase_time determines the month.



Explanation: Filters Oct-2020 + valid purchases, checks if any store has ≥5.

Output-No results (0 stores meets criteria)

Query 3 : Shortest interval to refund (in minutes) per store Assumptions Only rows with refund_item IS NOT NULL have valid refund timestamps. Refund_item column is assumed to be a datetime, not an item id → based on screenshot.Interval always = refund_time - purchase_time Negative / invalid values do not exist in the sample dataset.



Explanation: Uses TIMESTAMPDIFF to compute time difference in minutes.

Store f is fastest refund

Query 4 :Every store’s first order value "First order" means earliest purchase_time.If two transactions have the same timestamp for a store, the database ranking will choose one arbitrarily (not applicable in given data).

```
#Gross transaction value of each store’s first order
SELECT t.store_id, t.gross_transaction_value
FROM transactions t
JOIN (
    SELECT store_id, MIN(purchase_time) AS first_time
    FROM transactions
    GROUP BY store_id
) x ON t.store_id = x.store_id
AND t.purchase_time = x.first_time
ORDER BY t.store_id;
```

100% 7:68 2 errors found

Result Grid Filter Rows: Search Export:

store_id	gross_transaction_val...
a	58
b	475
b	475
d	250
e	24
f	33
g	61

Explanation: Subquery selects earliest purchase per store, joined back to details.

Query 5 : Most popular first-purchase item Assumptions: First purchase per buyer determined by earliest purchase_time. Only first-ever transaction per buyer is considered.If there is a tie, selecting the first in alphabetical/ordered result (not relevant in current dataset).

```
-- Most popular item name on buyers’ first purchase
WITH fp AS (
    SELECT buyer_id, item_id
    FROM transactions t
    WHERE purchase_time = (
        SELECT MIN(purchase_time)
        FROM transactions
        WHERE buyer_id = t.buyer_id
    )
)
SELECT i.item_name, COUNT(*) AS frequency
FROM fp
JOIN items i USING (item_id)
GROUP BY item_name
ORDER BY frequency DESC
LIMIT 1;
```

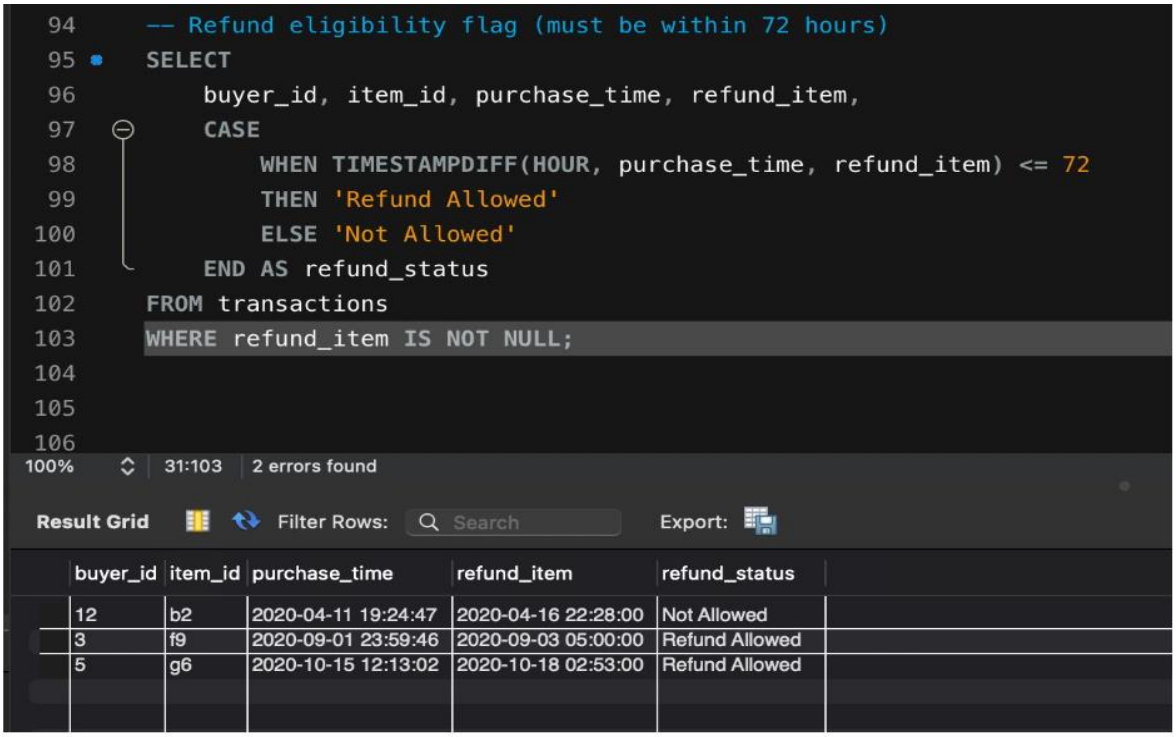
100% 9:91 2 errors found

Result Grid Filter Rows: Search Export:

item_name	frequency
shirt	2

Explanation: Finds earliest purchase for each buyer → joins item table → counts popularity.

Query 6 -Refund eligibility flag (within 72 hours) Assumptions: refund_item field stores refund timestamp (design confusion acknowledged).Hours difference measured using TIMESTAMPDIFF(HOUR, purchase_time, refund_item) Refund must happen within or equal to 72 hours to be eligible.If refund_item is NULL → “Not Valid”



```
94    -- Refund eligibility flag (must be within 72 hours)
95    SELECT
96        buyer_id, item_id, purchase_time, refund_item,
97        CASE
98            WHEN TIMESTAMPDIFF(HOUR, purchase_time, refund_item) <= 72
99            THEN 'Refund Allowed'
100           ELSE 'Not Allowed'
101       END AS refund_status
102   FROM transactions
103   WHERE refund_item IS NOT NULL;
104
105
106
```

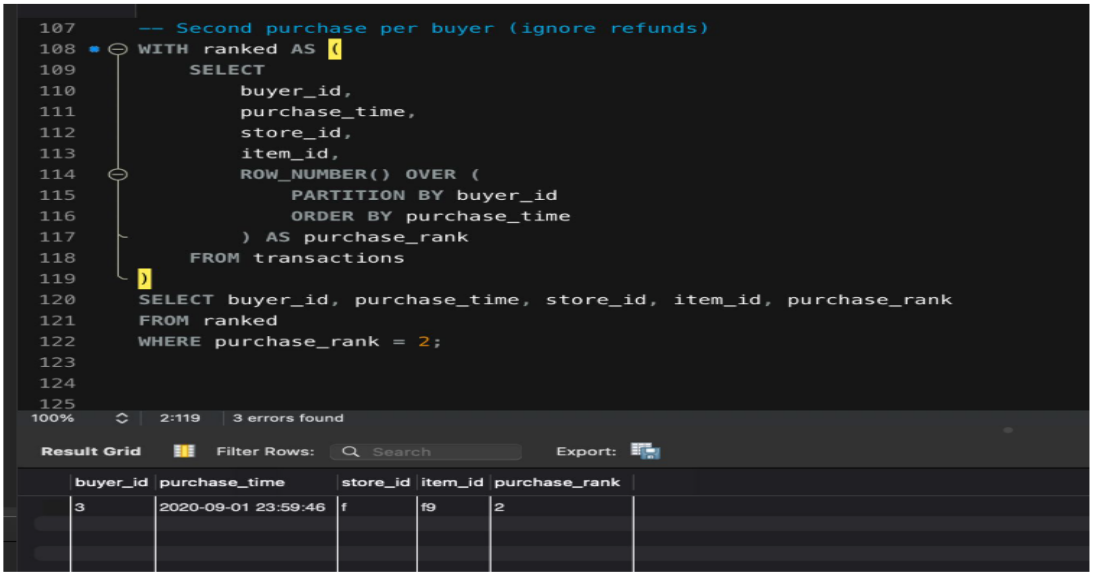
100% 31:103 2 errors found

Result Grid Filter Rows: Search Export:

buyer_id	item_id	purchase_time	refund_item	refund_status
12	b2	2020-04-11 19:24:47	2020-04-16 22:28:00	Not Allowed
3	f9	2020-09-01 23:59:46	2020-09-03 05:00:00	Refund Allowed
5	g6	2020-10-15 12:13:02	2020-10-18 02:53:00	Refund Allowed

Explanation: Refund processed only if refund time ≤ 72 hours.

Query 7 -Second purchase per buyer (ignoring refunds)Assumptions:Ranking is based solely on purchase_time.refund_item does not exclude a row from ranking.Only exactly the second chronological purchase returned.If a buyer has only 1 purchase → excluded from result.=Output expected: Only buyer_id = 3 has a second valid purchase.



```
107   -- Second purchase per buyer (ignore refunds)
108   WITH ranked AS (
109       SELECT
110           buyer_id,
111           purchase_time,
112           store_id,
113           item_id,
114           ROW_NUMBER() OVER (
115               PARTITION BY buyer_id
116               ORDER BY purchase_time
117           ) AS purchase_rank
118       FROM transactions
119   )
120   SELECT buyer_id, purchase_time, store_id, item_id, purchase_rank
121   FROM ranked
122   WHERE purchase_rank = 2;
123
124
125
```

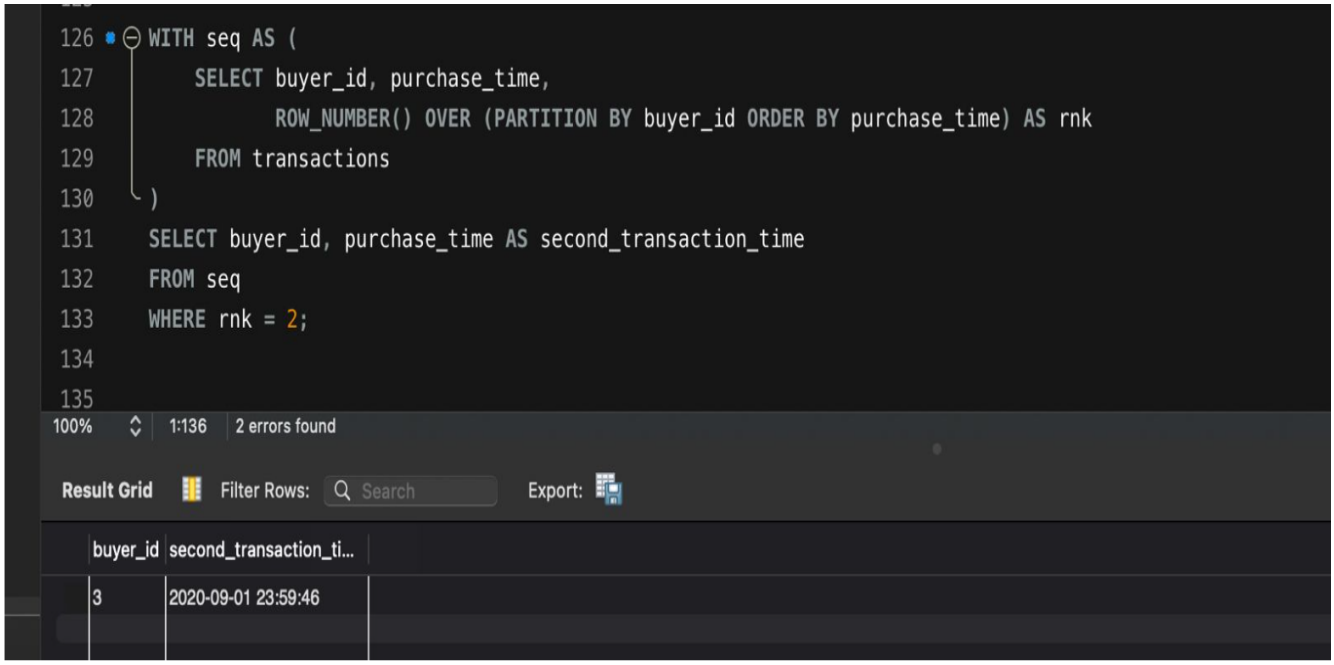
100% 2:119 3 errors found

Result Grid Filter Rows: Search Export:

buyer_id	purchase_time	store_id	item_id	purchase_rank
3	2020-09-01 23:59:46	f	f9	2

Explanation: Row-number assigns ranking per buyer based on purchase order.

Query 8-Find second transaction timestamp per buyer (without min/max) Same chronological rank logic as Q7, but using a counting subquery instead of window functions.A row is the second transaction if exactly one earlier transaction exists for that buyer.Again, only buyers with 2 or more purchases will appear.Expected output:Buyer 3's second purchase timestamp



-Not using min/max
-It includes refunded rows if they existed.