Text-to-SQL

Test Case of nature of natural language questions bitcoin data into an exact answer from executing on the database

NUID: 002030154/ Yu Ying Wu

10 Passed Test Cases

Natural Language Question	SQL statement	Actual Answer	Difficulty
What is the total block in the database	SELECT COUNT(*) FROM blocks;	486	Easy
what is the latest block height	SELECT MAX(height) FROM blocks;	861076	Easy
Is there duplicated nonce in the database	SELECT nonce, COUNT(*) FROM blocks GROUP BY nonce HAVING COUNT(*) > 1;	0	Easy
what is the average fee for all transactions	SELECT AVG(fee) AS average_price FROM transactions;	3.51506E-05	Easy
how many blocks timestamp are in Thursday, August 10, 2024	SELECT COUNT(*) FROM blocks WHERE DATE(timestamp, 'unixepoch') = '2024-08-10';	29	Medium
what is the number of transaction in block 856183	SELECT COUNT(*) FROM transactions WHERE block_id = (SELECT id FROM blocks WHERE height = 856183);	7162	Medium

what is the average price in Thursday, June 6, 2025	SELECT AVG(outputs.value) AS average_price
	FROM outputs
	JOIN transactions ON transactions.id = outputs.transaction_id
	JOIN blocks ON blocks.id = transactions.block_id
	WHERE DATE(blocks.created_at) = '2025-06-06';
		0.660939159956429	Difficult

what is the max price between 2025-06-05 to 2025-06-06	SELECT MAX(outputs.value) AS max_price FROM outputs JOIN transactions ON outputs.transaction_id = transactions.id JOIN blocks ON transactions.block_id = blocks.id WHERE datetime(blocks.created_at) BETWEEN '2025-06-05' AND '2025-06-06'	18907.70097322	Difficult
how many transaction in block 856183	SSELECT COUNT(*) FOM transactions WHERE block_id = (SELECT id FROM blocks WHERE height = 856183); FROM transactions WHERE block_id = (SELECT id FROM blocks WHERE height = 856183);	7162	Difficult
what is the average price in block 856183	SELECT AVG(outputs.value) AS average_price FROM outputs JOIN transactions ON transactions.id = outputs.transaction_id JOIN blocks ON blocks.id = transactions.block_id WHERE blocks.height = 856183;	0.0961991	Difficult

3 Failed Test Cases

<table>
<tr><th>Natural Language Question</th><th>Expected SQL statement</th><th>Expected Answer</th><th>Incorrect SQL Statement</th><th>Incorrect Answer</th></tr>
<tr>
<td>What is the average transaction fee per block for blocks that have more than 1000 transactions, and show the block height, timestamp, and total transaction count, but only for blocks where the average fee is higher than the overall average fee across all blocks? Order by block height descending and give me the Top 3?</td>
<td>WITH block_stats AS (
SELECT
b.height,
b.timestamp,
COUNT(t.txid) as tx_count,
AVG(t.fee) as avg_fee
FROM blocks b
JOIN transactions t ON b.id = t.block_id
GROUP BY b.height, b.timestamp
HAVING COUNT(t.txid) > 1000
),
overall_avg AS (
SELECT AVG(t.fee) as global_avg_fee
FROM transactions t
)
SELECT
bs.height,
datetime(bs.timestamp, 'unixepoch') as block_time,
bs.tx_count,
ROUND(bs.avg_fee, 8) as avg_fee_per_block
FROM block_stats bs
CROSS JOIN overall_avg oa
WHERE bs.avg_fee > oa.global_avg_fee
ORDER BY bs.height DESC
LIMIT 3</td>
<td>| Height | Block Time | TX Count | Avg Fee |
+==========+=====================+========
====+===========+
| 861792 | 2024-09-18 04:48:07 | 2277 | 3.372e-05 |
| 861098 | 2024-09-13 04:11:22 | 2463 | 3.314e-05 |
| 859730 | 2024-09-03 18:47:12 | 2649 | 4.685e-05 |</td>
<td>WITH block_fees AS (
SELECT
blocks.id AS block_id,
blocks.height AS block_height,
blocks.timestamp AS block_timestamp,
COUNT(transactions.id) AS transaction_count,
AVG(transactions.fee) AS avg_fee
FROM
blocks
INNER JOIN
transactions ON blocks.id = transactions.block_id
GROUP BY
blocks.id
HAVING
COUNT(transactions.id) > 1000
),
overall_avg_fee AS (
SELECT
AVG(avg_fee) AS avg_fee
FROM
block_fees
)
SELECT
block_height,
block_timestamp,
transaction_count,
avg_fee
FROM
block_fees
WHERE
avg_fee > (SELECT avg_fee FROM overall_avg_fee)
ORDER BY block_height DESC
Limit 3;</td>
<td>{'block_height': 859730, 'block_timestamp': 1725389232, 'transaction_count': 2649, 'avg_fee': 4.685011702529257e-05},
{'block_height': 859728, 'block_timestamp': 1725388792, 'transaction_count': 1703, 'avg_fee': 4.881324721080446e-05},
{'block_height': 859727, 'block_timestamp': 1725388413, 'transaction_count': 2771, 'avg_fee': 4.142688560086611e-05},</td>
</tr>
</table>

3 Failed Test Cases

Natural Language Question	Expected SQL statement	Expected Answer	Incorrect SQL Statement	Incorrect Answer
Find top 3 transactions that have at least 3 inputs and 3 outputs, where the total input value is more than 100 BTC, and show the transaction ID, total input value, total output value, and the difference between input and output values (fee). Also include the block height and timestamp. Only show transactions where the fee is in the top 10% of all transaction fees.	WITH tx_io_counts AS (SELECT t.txid, COUNT(DISTINCT i.input_id) as input_count, COUNT(DISTINCT o.output_id) as output_count, SUM(i.value) as total_input, SUM(o.value) as total_output, b.height, b.timestamp FROM transactions t JOIN inputs i ON t.txid = i.txid JOIN outputs o ON t.txid = o.txid JOIN blocks b ON t.blockhash = b.blockhash GROUP BY t.txid, b.height, b.timestamp HAVING input_count >= 3 AND output_count >= 3), fee_percentiles AS (SELECT txid, total_input, total_output, (total_input - total_output) as fee, height, timestamp, PERCENT_RANK() OVER (ORDER BY (total_input - total_output)) as fee_percentile FROM tx_io_counts WHERE total_input > 10000000000 -- 100 BTC in satoshis) SELECT txid, ROUND(total_input / 100000000.0, 8) as total_input_btc, ROUND(total_output / 100000000.0, 8) as total_output_btc, ROUND(fee / 100000000.0, 8) as fee_btc, height, datetime(timestamp, 'unixepoch') as block_time FROM fee_percentiles WHERE fee_percentile >= 0.9 ORDER BY fee DESC LIMIT 3;	No results	WITH transaction_values AS (SELECT transactions.txid, blocks.height, blocks.timestamp, SUM(inputs.value) AS total_input_value, SUM(outputs.value) AS total_output_value, (SUM(inputs.value) - SUM(outputs.value)) AS fee FROM transactions JOIN blocks ON transactions.block_id = blocks.id JOIN inputs ON transactions.id = inputs.transaction_id JOIN outputs ON transactions.id = outputs.transaction_id GROUP BY transactions.txid HAVING COUNT(inputs.id) >= 3 AND COUNT(outputs.id) >= 3 AND SUM(inputs.value) > 10000000000), transaction_fees AS (SELECT fee FROM transaction_values ORDER BY fee DESC LIMIT (SELECT COUNT(*) FROM transaction_values) / 10) SELECT txid, total_input_value, total_output_value, fee, height, timestamp FROM transaction_values WHERE fee IN transaction_fees ORDER BY fee DESC LIMIT 3;	No result

3 Failed Test Cases

Natural Language Question	Expected SQL statement	Expected Answer	Incorrect SQL Statement	Incorrect Answer

Find 3 blocks where the total transaction volume (sum of all input values) is more than 1000 BTC, and for each of these blocks, show the number of unique addresses that received outputs, the average output value per address, and the percentage of outputs that were sent to addresses that appear more than once in the block. Also include the block height and timestamp.	SELECT
	    b.height,
	    b.timestamp,
	    COUNT(DISTINCT o.address) AS unique_addresses,
	    AVG(o.value) AS avg_output_value,
	    (COUNT(o.address) - COUNT(DISTINCT o.address)) * 100.0 / COUNT(o.address) AS percentage_repeat_addresses
	FROM
	    blocks b
	JOIN
	    transactions t ON b.id = t.block_id
	JOIN
	    inputs i ON t.id = i.transaction_id
	JOIN
	    outputs o ON t.id = o.transaction_id
	GROUP BY
	    b.id
	HAVING
	    SUM(i.previous_vout) > 1000
	LIMIT 3;
	Executing query: SELECT
	    b.height,
	    b.timestamp,
	    COUNT(DISTINCT o.address) AS unique_addresses,
	    AVG(o.value) AS avg_output_value,
	    (COUNT(o.address) - COUNT(DISTINCT o.address)) * 100.0 / COUNT(o.address) AS percentage_repeat_addresses
	FROM
	    blocks b
	JOIN
	    transactions t ON b.id = t.block_id
	JOIN
	    inputs i ON t.id = i.transaction_id
	JOIN
	    outputs o ON t.id = o.transaction_id
	GROUP BY
	    b.id
	HAVING
	    SUM(i.previous_vout) > 1000
	LIMIT 3;
		[{'height': 845993, 'timestamp': 1717204315, 'unique_addresses': 0, 'avg_output_value': 0.24576988061685112, 'percentage_repeat_addresses': None}, {'height': 846136, 'timestamp': 1717297249, 'unique_addresses': 0, 'avg_output_value': 0.0406693117305406, 'percentage_repeat_addresses': None}, {'height': 846137, 'timestamp': 1717299941, 'unique_addresses': 0, 'avg_output_value': 0.406662864922617, 'percentage_repeat_addresses': None}]	WITH block_volumes AS (
			        SELECT
			            b.height,
			            b.timestamp,
			            SUM(o.value) as total_volume
			        FROM blocks b
			        JOIN transactions t ON b.id = t.block_id
			        JOIN outputs o ON t.id = o.transaction_id
			        GROUP BY b.height, b.timestamp
			        HAVING SUM(o.value) > 100000000000 -- 1000 BTC in satoshis
			    ),
			    address_stats AS (
			        SELECT
			            b.height,
			            COUNT(DISTINCT o.address) as unique_addresses,
			            AVG(o.value) as avg_output_per_address,
			            COUNT(CASE WHEN address_count > 1 THEN 1 END) * 100.0 / COUNT(*) as percent_repeated_addresses
			        FROM blocks b
			        JOIN transactions t ON b.id = t.block_id
			        JOIN outputs o ON t.id = o.transaction_id
			        JOIN (
			            SELECT height, address, COUNT(*) as address_count
			            FROM blocks b2
			            JOIN transactions t2 ON b2.id = t2.block_id
			            JOIN outputs o2 ON t2.id = o2.transaction_id
			            GROUP BY height, address
			        ) addr_counts ON b.height = addr_counts.height AND o.address = addr_counts.address
			        WHERE b.height IN (SELECT height FROM block_volumes)
			        GROUP BY b.height
			    )
			    SELECT
			        bv.height,
			        datetime(bv.timestamp, 'unixepoch') as block_time,
			        ROUND(bv.total_volume / 100000000.0, 8) as total_volume_btc,
			        addr_stats.unique_addresses,
			        ROUND(addr_stats.avg_output_per_address / 100000000.0, 8) as avg_output_per_address_btc,
			        ROUND(addr_stats.percent_repeated_addresses, 2) as percent_repeated_addresses
			    FROM block_volumes bv
			    JOIN address_stats addr_stats ON bv.height = addr_stats.height
			    ORDER BY bv.height DESC
			    LIMIT 3;
				No results