

Your first wand on the path to Dune wizardry

Query Skeleton

The fundamental structure of every query

Query Execution Order



Core Structure

Every clause has a specific purpose in your query

SELECT	What to return Choose specific columns or calculations
FROM	Which table to use Specify the data source for your query
WHERE	Filter by condition Remove unwanted rows from results
GROUP BY	Group rows into buckets Combine rows for aggregation functions
HAVING	Filter after grouping Keep only groups that meet an aggregate condition
ORDER BY	Sort the output Control the sequence of returned rows
LIMIT	Control row count Limit the number of results returned

Blockchain Example

A real query showing how all components work together

Largest ETH Senders in the Last 7 Days

```

SELECT
  "from" AS sender_wallet,
  SUM(value / 1e18) AS total_eth_sent
FROM ethereum.transactions
WHERE
  block_time >= now() - interval '7' day
GROUP BY
  "from"
HAVING
  SUM(value / 1e18) > 10
ORDER BY
  total_eth_sent DESC
LIMIT 10
  
```

Functions

Your toolbox for doing math and creating metrics

Aggregation Functions

Summarize data across multiple rows

COUNT()	Number of rows	
SUM()	Total value	
AVG()	Average value	
MIN()	MAX()	Find extremes in data

Math Helpers

Transform and format numerical data

ROUND(x,d)	Round to d decimals Control precision display
FLOOR(x)	Round down Always rounds down
CEIL(x)	Round up Always rounds up
ABS(x)	Absolute value Remove negative sign

Text Functions

Transform and manipulate text data

LOWER(text)	Converts to lowercase
UPPER(text)	Converts to uppercase
SUBSTR (string, start, length)	Extract text
CONCAT (first text, last text)	Combine texts

CASE WHEN
creates categories →



```

CASE
  WHEN amount < 10 THEN 'Small'
  WHEN amount < 100 THEN 'Medium'
  ELSE 'Large'
END AS size_group
  
```

*Time functions covered in next section

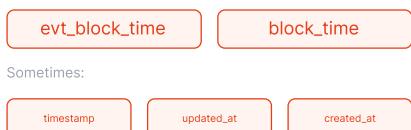
Working With Time

How to turn timestamps into usable time series

3 Step Time Series Recipe

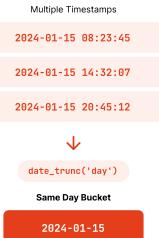
1 Find the Timestamp

The column that shows when events happened



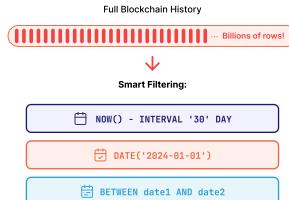
2 Bucket Into Periods

Use date_trunc to group timestamps



3 Filter Time Range

Keep only the time window you want



Pro Tip: Filling Empty Days

Use `utils.{time period}` with a LEFT JOIN to produce a complete timeline.

Joins

Everything you need to combine tables into more complete datasets



Common Dune Tables to Join

• Token Metadata

`tokens.erc20`
Join on: contract_address and blockchain
Use for: symbol, name, decimals

• Prices

`prices.{time period}`
Join on: contract_address and date_trunc() aligned timestamps
Use for: USD values, TVL, volume

• Labels

`labels.addresses`
Join on: address and blockchain
Use for: labeling addresses

• Transactions

`ethereum.transactions`
Join on: block_time
Use for: gas, sender/receiver

• DEX data

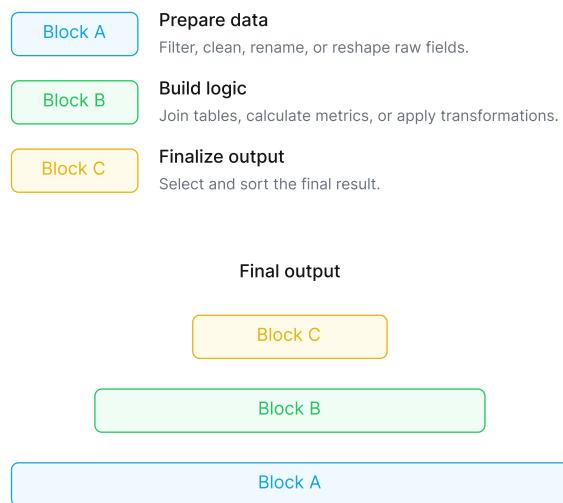
`dex.trades`
Join on: project_contract_address
Use for: trading activities

WITH common_table_expression also known AS CTEs

A clean way to break long SQL queries into readable blocks

How CTEs Work

Think of CTEs as building blocks. Each block can do anything: filtering, reshaping, joining, aggregating, or calculating.



Blockchain Example

A real query showing how all components work together

```

WITH daily_volumes AS (
  SELECT
    block_date as trade_date,
    SUM(amount_usd) as daily_volume
  FROM dex.trades
  WHERE blockchain = 'ethereum'
    AND block_date >= TIMESTAMP '2024-09-01'
    AND block_date < TIMESTAMP '2024-10-01'
  GROUP BY block_date
),
volume_metrics AS (
  SELECT
    trade_date,
    daily_volume,
    AVG(daily_volume) OVER (
      ORDER BY trade_date
      ROWS BETWEEN 6 PRECEDING AND CURRENT
    ) as rolling_7day_avg
  FROM daily_volumes
)
SELECT * FROM volume_metrics
ORDER BY trade_date DESC
  
```

{{Parameters}}

Make your queries and dashboards interactive by letting users choose values that your DuneSQL reacts to

`{{Text Parameter}}`

`{{Number Parameter}}`

`{{Date Parameter}}`

List Parameter

One or more options from a dropdown

Manual List

Create static dropdowns

List From Other Query Results

Build dynamic dropdowns

Use in chart titles

e.g. "Activity in the last {{Days}} days"

Shared Parameters

Use the same parameter across multiple queries by matching name, type, default, and list settings exactly

Window Functions

Lets you add extra columns like ranks or rolling averages without changing the rows you already have

Window Setup
FUNCTION adds a calculation, **OVER** applies it across rows, **PARTITION BY** groups them, **ORDER BY** sets the order inside each group.

Syntax

```

FUNCTION(column) OVER (
  PARTITION BY group_column
  ORDER BY sort_column) AS
column_name
)
  
```

Ranking

Add row order

`ROW_NUMBER()`
`RANK()`
`DENSE_RANK()`

Time Shifts

See previous or next values

`LAG()`
`LEAD()`

Rolling Metrics

Create moving totals or averages

`SUM()`
`AVG()`
`MIN() / MAX()`

Decoding Blockchain Events

How to find, read, and request decoded contract data on Dune



Working With API, JSON & Array Data

Everything you need to turn messy or external data into usable tables

