06. Useful Queries — From Token Transfers to Whale Watching

Once you're comfortable with SQL, the next step is mastering the right questions to ask.

Onchain data is rich—but messy. The best analysts learn to cut through the noise and surface signal. In this article, we'll share a collection of **useful query templates** for everyday analysis —whether you're building dashboards, writing reports, or just exploring.

Each example below includes the query goal, key tables, and a reusable SQL snippet.

 Token Transfers

Goal: Track ERC20 transfers by volume or frequency.

Table: `erc20.token_transfers`

```
SELECT
  date_trunc('day', block_time) AS day,
  COUNT(*) AS transfer_count,
  SUM(amount / 1e18) AS token_volume
FROM erc20.token_transfers
WHERE token_address = LOWER('0xA0b86991c6218b36c1d19D4a2e9Eb0cE3606eB48') -
- USDC
  AND block_time > now() - interval '30 days'
GROUP BY 1
ORDER BY 1
```

🧠 Tip: Use `LOWER()` for all token/wallet addresses to ensure case-insensitive matching.

🧟 NFT Wash Trading Detection

Goal: Spot suspicious NFT sales to self or known alt wallets.

Table: `nft.trades`

```
SELECT
  buyer,
  seller,
  COUNT(*) AS trades,
  SUM(price_usd) AS volume
FROM nft.trades
WHERE buyer = seller
  AND block_time > now() - interval '30 days'
GROUP BY 1, 2
ORDER BY 4 DESC
```

🧠 Tip: Filter trades where buyer = seller or use wallet clustering to flag sybil behavior.

🏦 DAO Treasury Monitoring

Goal: Track stablecoin inflows/outflows to a DAO-controlled multisig.

Table: `erc20.token_transfers`

```
SELECT
  date_trunc('day', block_time) AS day,
  SUM(CASE WHEN to_address = LOWER('0xDAOwallet') THEN amount / 1e6 ELSE 0
END) AS inflow,
  SUM(CASE WHEN from_address = LOWER('0xDAOwallet') THEN amount / 1e6 ELSE
0 END) AS outflow
FROM erc20.token_transfers
WHERE token_address = LOWER('0xdAC17F958D2ee523a2206206994597C13D831ec7') -
- USDT
  AND block_time > now() - interval '60 days'
GROUP BY 1
ORDER BY 1
```

🧠 Whale Watching

Goal: Identify the top ERC20 recipients in the past week.

Table: `erc20.token_transfers`

```
SELECT
  to_address,
  COUNT(*) AS tx_count,
  SUM(amount / 1e18) AS received_tokens
FROM erc20.token_transfers
WHERE block_time > now() - interval '7 days'
GROUP BY 1
ORDER BY 3 DESC
LIMIT 10
```

🧠 Tip: Use this with labels or merge with wallet tag datasets to identify CEXs, whales, or teams.

📈 DEX Swap Volume by Token

Goal: Analyze swap volume for a specific token on Uniswap V3.

Table: `uniswap_v3.uniswap_v3_swaps`

```
SELECT
  date_trunc('day', block_time) AS day,
  SUM(amount_usd) AS daily_volume
FROM uniswap_v3.uniswap_v3_swaps
WHERE (token_in = LOWER('0x...') OR token_out = LOWER('0x...')) -- Insert token
  AND block_time > now() - interval '14 days'
GROUP BY 1
ORDER BY 1
```

💸 Protocol Fee Revenue

Goal: Measure revenue accrued by protocols with fee switches enabled.

Table: protocol-specific (e.g., `uniswap_v3.fees`, `aave.liquidations`, etc.)

```
SELECT
  date_trunc('day', block_time) AS day,
  SUM(fee_amount_usd) AS revenue
FROM uniswap_v3.fees
WHERE block_time > now() - interval '30 days'
GROUP BY 1
ORDER BY 1
```

🧠 Tip: Use `prices.usd` to normalize gas or token fees to USD for consistency.

🛠️ Cross-Chain Activity (Spellbook)

If you're using Spellbook models from DuneSQL, try this unified userop query:

```
SELECT
  blockchain,
  COUNT(*) AS ops,
  SUM(op_fee_usd) AS total_fees
FROM account_abstraction_erc4337.userops
WHERE block_time > now() - interval '14 days'
GROUP BY 1
ORDER BY 3 DESC
```

📊 Dashboard Building Patterns

- Use `date_trunc()` for time series
- Normalize to USD when comparing tokens
- Always use `LIMIT` when exploring large tables
- Comment your code for future you (and others!)

- Combine queries with CTEs to keep logic clean

🧭 Where to Find More Queries

- Browse the Dune Discover page
- Explore the Spellbook GitHub repo
- Fork dashboards from top analysts (and reverse-engineer!)
- Join Telegram/Discord communities and ask questions

Your Onchain Toolbox Grows

As you start saving useful queries, you'll build your own **playbook** of blockchain insights.

Eventually, you'll recognize patterns before writing code:

- Airdrop season? Check claim activity and sybil patterns.
- New NFT mint? Track mint velocity and secondary flips.
- Yield spike? Trace source of new liquidity and associated risk.

Next: 07. NFT Analysis — Wash Trading, Mint Trends, and Market Health