

# ClickHouse Analytics Platform

## Quick Reference Guide

We have recently moved to ClickHouse to handle Analytics. This document provides a brief overview.

Feature	PostgreSQL (Row-Based)	ClickHouse (Column-Based)
Storage Model	Row by row	Column by column
Optimization	OLTP (transactional)	OLAP (analytical)
Best For	Updates, deletes, complex transactions	Large scans and aggregations
Indexes	B-tree on every row	Sparse (1 per 8,192 rows)
Workload	Mutable data	Append-only, immutable
Data Lag	Real-time	~1 minute from source

### Key Points

- ClickHouse is very fast for aggregations
- Data from both Jobfinder and Warehouse\_RDS is available
- You can join tables across different schemas like in PostgreSQL
- There is approximately 1 minute lag between source databases and ClickHouse

**Example Query:** Count calls from October 1-20, 2025

```
select count(1) from recruiter_call_logs where  
"startedAt">'2025-10-01'  
and "startedAt"<'2025-10-20'
```

Database	Execution Time
PostgreSQL (Jobfinder)	16 seconds
ClickHouse	1 second (16x faster!)

## 1. NEVER Use SELECT \*

ClickHouse reads data column by column. Every column you select adds I/O overhead.

### ✗ DON'T DO THIS:

```
SELECT *  
FROM jobfinder.recruiter_call_logs  
WHERE "startedAt"<'2025-10-20'
```

### ✓ DO THIS INSTEAD:

```
SELECT "callerNumber", "receiverNumber", "duration"  
FROM jobfinder.recruiter_call_logs  
WHERE "startedAt"<'2025-10-20'
```

**Impact:** 5-20x faster

## 2. Use PREWHERE Instead of WHERE

**WHERE:** Reads ALL columns from disk, then filters

**PREWHERE:** Reads filter columns first, then only reads remaining columns for matching rows

```
SELECT "callerNumber", "receiverNumber", "duration"  
FROM jobfinder.recruiter_call_logs  
PREWHERE "startedAt"<'2025-10-20'
```

**Impact:** 2-5x faster

## 3. Use uniqExact() Instead of COUNT(DISTINCT)

More efficient for counting unique values.

```
SELECT uniqExact(distinct("id"))
FROM jobfinder.recruiter_call_logs
PREWHERE "startedAt">'2025-10-19'
AND "startedAt"<'2025-10-20'
```

**Impact: 5-10x faster**

## 4. JOIN Best Practices

- **Always ensure the smallest table is on the RIGHT side of the JOIN**
- The right table is loaded into memory, so keep it small
- Filter BEFORE joining to reduce rows
- Use IN instead of INNER JOIN when possible for simple lookups

### Example: Less Efficient (11 seconds)

```
SELECT uniqExact(co."phone_number")
FROM jobfinder.candidates_olap co
JOIN jobfinder.recruiter_call_logs rcl
  ON rcl."receiverNumber" = co."phone_number"
PREWHERE rcl."createdAt">'2025-01-01'
SETTINGS use_query_cache = 0
```

### Example: More Efficient (4 seconds)

```
SELECT uniqExact(co."phone_number")
FROM jobfinder.candidates_olap co
PREWHERE "phone_number" IN (
  SELECT "receiverNumber"
  FROM jobfinder.recruiter_call_logs rcl
  WHERE rcl."createdAt">'2025-01-01'
)
SETTINGS use_query_cache = 0
```

## 5. String Matching: hasToken() vs LIKE

### Use hasToken() when:

- Searching for complete words/tokens
- String has natural delimiters (spaces, underscores, dots)
- You need "word boundary" matching

### Use LIKE when:

- You need partial word matching

- Pattern is in the middle of a word
- You need complex pattern matching with % wildcards

### Slower (LIKE):

```
SELECT uniqExact("id")
FROM jobfinder.recruiter_call_logs
PREWHERE "startedAt">'2025-01-01'
AND "startedAt"<'2025-10-20'
AND "tag" LIKE '%Service%'
```

### Faster (hasToken):

```
SELECT uniqExact("id")
FROM jobfinder.recruiter_call_logs
PREWHERE "startedAt">'2025-01-01'
AND "startedAt"<'2025-10-20'
AND hasToken("tag", 'Service')
```

**Impact:** 10-50x faster for word matching

PostgreSQL to ClickHouse function mapping:

PostgreSQL	ClickHouse	Description
DATE_TRUNC('day', ts)	toStartOfDay(ts)	Start of day
DATE_TRUNC('month', ts)	toStartOfMonth(ts)	Start of month
DATE_TRUNC('hour', ts)	toStartOfHour(ts)	Start of hour
EXTRACT(YEAR FROM ts)	toYear(ts)	Extract year
EXTRACT(MONTH FROM ts)	toMonth(ts)	Extract month
EXTRACT(DAY FROM ts)	toDayOfMonth(ts)	Extract day
EXTRACT(DOW FROM ts)	toDayOfWeek(ts)	Day of week (1-7)
COUNT(DISTINCT id)	uniq(id)	Count unique (approx)

PostgreSQL	ClickHouse	Description
<code>COUNT(DISTINCT id)</code>	<code>uniqExact(id)</code>	Count unique (exact)

Operation	PostgreSQL Way	ClickHouse Way
<b>Count distinct</b>	<code>COUNT(DISTINCT id)</code>	<code>uniq(id)</code>
<b>Select all columns</b>	<code>SELECT * acceptable</code>	<b>Never use</b>
<b>Filtering</b>	<code>WHERE</code>	<b>PREWHERE</b>
<b>String search</b>	<code>LIKE '%text%'</code>	<code>hasToken()</code> or <code>position()</code>
<b>Multiple JOINS</b>	Optimized	<b>Avoid if possible (denormalize)</b>