TIME SERIES DATA WITH SQL

```
SELECT
   time_bucket('1 hour', time) AS hour,
   device_id,
   AVG(temperature) AS avg_temp
FROM
    readings
WHERE
   time  > NOW() - interval "24 hours"
   GROUP BY: 1, 2
   ORDER BY:.
           hour DESC;
```

Rajnish Kumar
Data Architect

Content *overview*

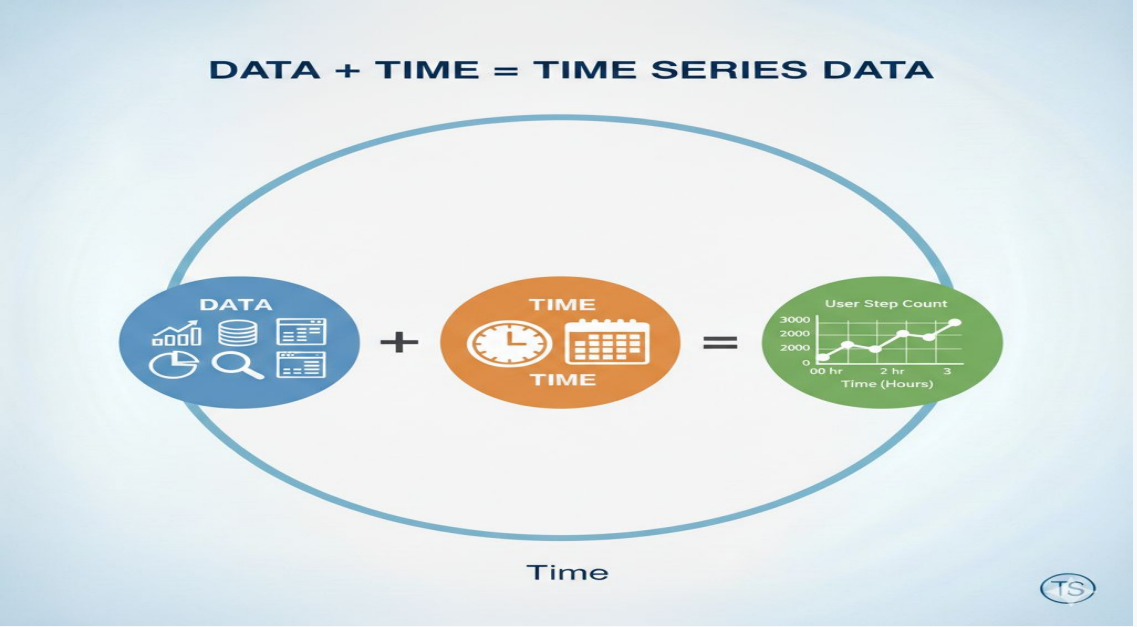
What is Time Series Data?

Time series data is a sequence of data points collected, recorded, or indexed in time order.

Each data point is associated with a timestamp, creating a chronological sequence.

For Quick Understanding :

Data + Time = Time Series Data



Key Challenges

Data Quality & Signal Integrity

- **Missing Data:** Sensor failures and ingestion issues create incomplete time series
- **Noise & Outliers:** Random disturbances and spikes distort analysis and forecasting accuracy

Scale & Performance

- **High Volume & Velocity:** Millions of events per second require scalable ingestion and storage
- **High Cardinality:** Large dimensional datasets degrade indexing, storage efficiency, and query performance

Temporal Complexity

- **Non-Stationarity (Concept Drift):** Data patterns evolve, requiring continuous model adaptation

- **Time Alignment & Late Arrivals:** Distributed data sources introduce synchronization and ordering challenges

Operational & Cost Constraints

- **Query Latency:** Time-window aggregations and long-range scans are computationally intensive

- **Retention & Downsampling:** Balancing historical accuracy with storage and compute cost

- **Real-Time vs Historical Processing:** Dual processing paths increase system complexity and operational overhead

What Is TimescaleDB ?

TimescaleDB is an open-source, time-series database that is implemented as an extension of PostgreSQL. It is designed to combine the reliability, full SQL support, and rich ecosystem of a traditional relational database with the scalability and performance optimizations required for handling massive volumes of time-stamped data.

Key Features

- **PostgreSQL Compatibility: Because it runs as an extension, TimescaleDB uses standard SQL, meaning developers can use existing PostgreSQL tools, libraries, and ORMs without learning a new query language (unlike some other TSDBs).**
- **Hypertables & Chunking: This is the core architectural feature. From a user's perspective, you interact with a single "hypertable," which functions like a normal SQL table. Internally, TimescaleDB automatically partitions this data into smaller, manageable physical tables called "chunks" based on time (and optionally other dimensions), which drastically improves query speed and data ingestion performance.**
- **Compression: TimescaleDB offers native, columnar compression for older, less frequently accessed data. This can reduce storage size by over 90% while still keeping the data queryable.**

- **Continuous Aggregates: This feature automatically creates and maintains materialized views of pre-computed, summarized data (e.g., hourly averages of sensor readings). This makes dashboard queries and long-term analytics run much faster.**
- **Data Retention Policies: It provides built-in functions to automatically manage the data lifecycle, such as dropping old chunks of data efficiently without impacting performance.**

TimescaleDB Deployment Options

Self-Hosted Deployment

Your Infrastructure

- Full Control
- Custom Configuration
- On-Premise Hardware

Managed Cloud Service

Timescale Cloud

- Fully Managed
- Automated Scaling
- 24/7 Support

When TimescaleDB Is a Strong Choice

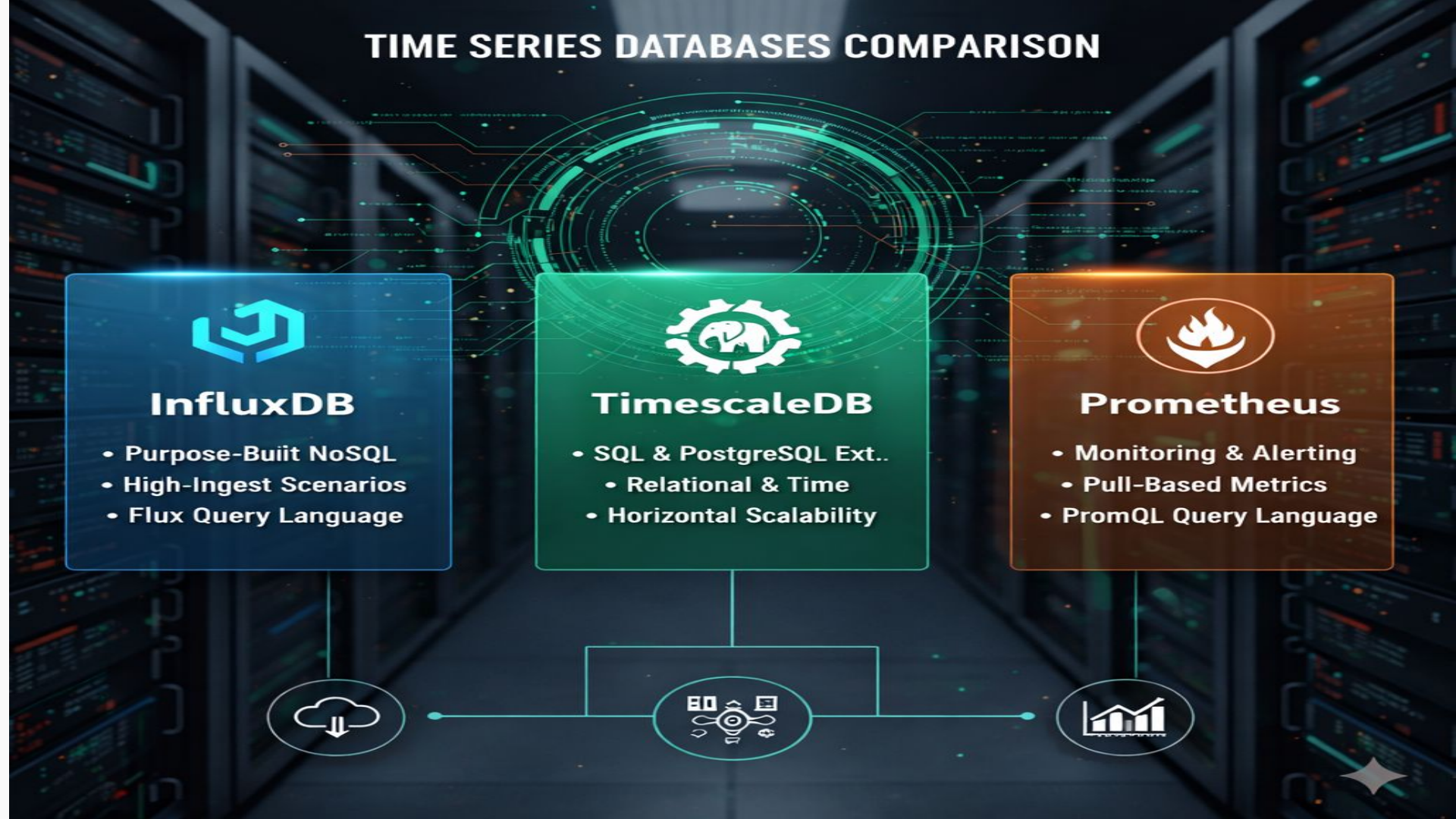
- **Existing PostgreSQL Adoption**
  Leverages current PostgreSQL infrastructure, tools, and skills

- **SQL-First Time Series Analytics**
  Enables advanced analytics using standard SQL without learning new query languages

- **Hybrid OLTP + Time Series Workloads**
  Supports transactional and time-series data within a single platform

- **Strong Ecosystem Integration**
  Seamlessly integrates with PostgreSQL extensions, BI tools, and monitoring platforms

TIME SERIES DATABASES COMPARISON



InfluxDB

- Purpose-Buiit NoSQL
- High-Ingest Scenarios
- Flux Query Language

TimescaleDB

- SQL & PostgreSQL Ext..
- Relational & Time
- Horizontal Scalability

Prometheus

- Monitoring & Alerting
- Pull-Based Metrics
- PromQL Query Language

Quick Comparison

Feature	TimescaleDB	InfluxDB	Prometheus
	✓ Full SQL	✓ Limited	✗ None
SQL Support	✓ Yes	✗ Yes	✗ None
Relational Data	✓ Yes	✗ No	✗ No
Monitoring Focus	✓ Yes	✓ Yes	✓ Strong
Long-Term Storage	✓ Strong	✓ Strong	⚠ Limited

Conclusion :

- **TimescaleDB** stands out as the most versatile option for general-purpose time-series data management. It provides full SQL compatibility and support for relational data, making it ideal if you need a standard database experience while handling time-series workloads. It also offers strong capabilities for long-term storage and monitoring focus.
- **InfluxDB** provides a strong alternative for scenarios where full relational data support is not a requirement. It balances monitoring focus with robust long-term storage capabilities, though its SQL support is limited compared to TimescaleDB.
- **Prometheus** is highly specialized for monitoring specific infrastructure metrics. Its lack of SQL support and limited long-term storage capabilities mean it is best suited for temporary, real-time monitoring and alerting systems rather than deep historical analytics.

The best choice depends on whether you prioritize **full SQL support and relational structure** (TimescaleDB) or a more specialized, potentially simpler monitoring stack (InfluxDB or Prometheus).

TimescaleDB

Full SQL & Relational Data
Versatile Time-Series DB

InfluxDB

Purpose-Built for Monitoring
Balanced Long-Term Storage

Promethus

Specialized Metrics & Alerts
Real-Time Monitoring Stack

CHOOSE YOUR TIME-SERIES PLATFORM

SQL? Monitoring? Both?

Thank You!

Thank you for your time and attention.

Rajnish Kumar

Data Architect