


YUGANK SRIVASTAVA

Brightly

# "AI-Assisted Post-Mortem Analysis and Optimizations for PostgreSQL Performance"

# AI-Assisted Post-Mortem Analysis and Optimizations for PostgreSQL Performance

Explore how AI revolutionizes PostgreSQL performance tuning. This presentation covers AI's role in post-mortem analysis. Learn about optimizing database performance proactively.

# ABOUT ME

- Database professional with 9+ years of experience.
- Expertise in cloud databases practices across AWS, Azure.
- Proficient in automating database operations and scalability.
- Strong interest in performance tuning of DB's in cloud environments.

## My Career Journey

1 **IBM**

core database skills foundation.

2 **Gartner**

cloud based database expertise

3 **Brightly**

exploring new challenges within cloud databases.

# AGENDA


- What is Post-Mortem Analysis in Databases?
- The Need for AI in Database Performance: Why Now?
- AI-Assisted Post-Mortem Analysis Workflow
- PostgreSQL Performance Bottlenecks: A Review
- Introducing the AI Analyst: Architecture and Capabilities
- Data Collection and Feature Engineering for AI-Driven Insights
- Anomaly Detection with AI: Identifying the Root Cause
- Predictive Optimization: Proactive Tuning Recommendations
- Tools and Technologies for AI-Assisted Analysis
- AI-Powered PostgreSQL Performance Boost: An E-Commerce Success Story
- Revolutionizing Efficiency: Exploring AI-Driven Optimization in PostgreSQL
- DEMO: Create and Analyze a Blocking Transaction
- Conclusion: Unlock PostgreSQL Performance with AI
- Questions and Answers


# What is Post-Mortem Analysis in Databases?

1

## Incident Review

Analyzing past database incidents is vital for maintaining system health and reliability. This systematic review process examines system logs, performance metrics, and user reports to determine the root causes of failures or slowdowns. Teams document the timeline of events, impact on users, and technical factors that contributed to the incident.

2

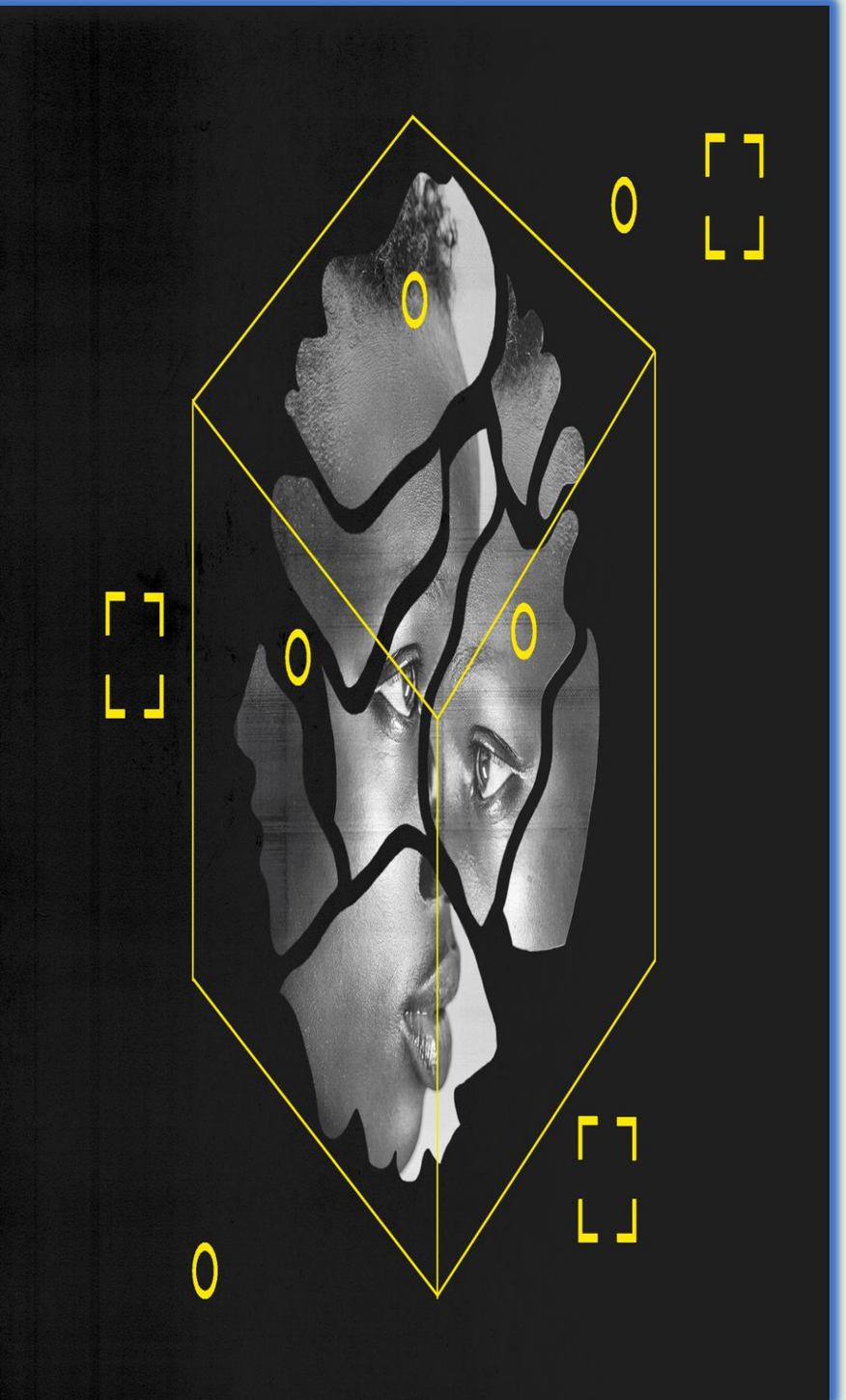
## Performance Gaps

Identify performance bottlenecks and systemic weaknesses through detailed analysis of historical data. This includes examining query execution plans, resource utilization patterns, and infrastructure metrics to understand patterns leading to failures. Performance gaps might manifest as slow query response times, memory pressure, or I/O bottlenecks that impact database operations.

3

## Optimization

Improve future database performance through targeted optimizations based on post-mortem findings. This involves implementing better indexing strategies, query optimization, and resource allocation adjustments. Teams create action plans to prevent similar incidents from occurring, establish monitoring alerts, and develop best practices for database maintenance and scaling.



# *The Need for AI in Database Performance: Why Now?*

## Growing Data Volumes

Data is growing exponentially. Traditional methods are struggling to keep up.

## Complex Systems

Modern databases are intricate. AI excels at navigating this complexity.

## Need for Speed

Real-time insights are crucial. AI offers faster analysis and tuning.

# AI-Assisted Post-Mortem Analysis Workflow

1

## Data Collection

Systematically gather comprehensive data from the database system, including performance logs, query execution metrics, resource utilization statistics, and user-reported issues. This forms the foundation for accurate analysis.

2

## AI Analysis

Advanced AI algorithms process the collected data using machine learning techniques to detect patterns, identify performance anomalies, and establish correlations between different metrics. This automated analysis helps surface insights that might be missed by traditional methods.

3

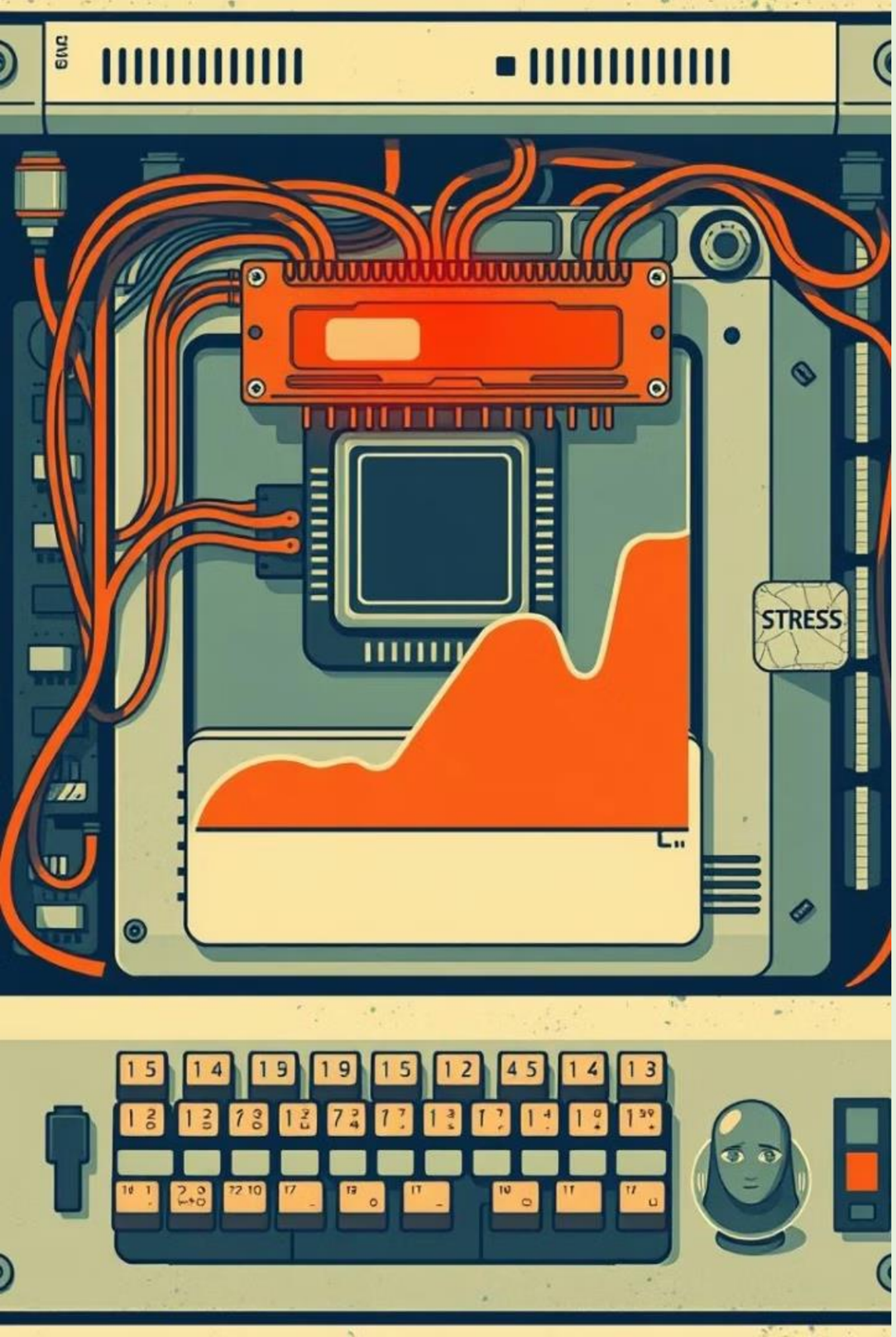
## Root Cause Investigation

Through sophisticated pattern recognition and causal analysis, AI pinpoints the fundamental source of performance issues. This step goes beyond surface-level symptoms to uncover hidden bottlenecks, configuration problems, and systemic weaknesses in the database architecture.

4

## Implementation & Optimization

Based on the AI's findings, implement targeted performance tuning and strategic optimizations. This includes query rewrites, index adjustments, configuration changes, and architectural improvements. Monitor the results to ensure the solutions effectively prevent similar issues from recurring.




# PostgreSQL Performance Bottlenecks: A Review

## Slow Queries

Inefficient SQL queries significantly degrade database performance through excessive resource consumption and longer execution times. Common issues include missing indexes, poor join conditions, and suboptimal query patterns. Optimize queries through proper indexing strategies, query restructuring, and regular EXPLAIN ANALYZE evaluation.

## Lock Contention

Excessive locking mechanisms create bottlenecks by forcing concurrent transactions to wait unnecessarily. This leads to increased response times and reduced throughput. Reduce contention through proper transaction management, optimized isolation levels, and strategic approach to table partitioning. Consider implementing row-level locking where appropriate.

## Resource Limits

Insufficient memory allocation forces PostgreSQL to rely more heavily on disk operations, creating significant performance bottlenecks. Key settings like shared_buffers, work_mem, and effective_cache_size need careful tuning. Monitor memory usage patterns and adjust resource allocation based on workload characteristics and available system resources.

## Disk I/O

Slow disk I/O operations create a severe performance bottleneck, particularly during heavy write operations or when working with large datasets. Improve storage configuration through proper RAID setup, SSD implementation, and strategic placement of WAL logs. Consider implementing proper vacuum strategies and monitoring I/O patterns to optimize disk usage.

# Introducing the AI Analyst: Architecture and Capabilities

### Anomaly Detection

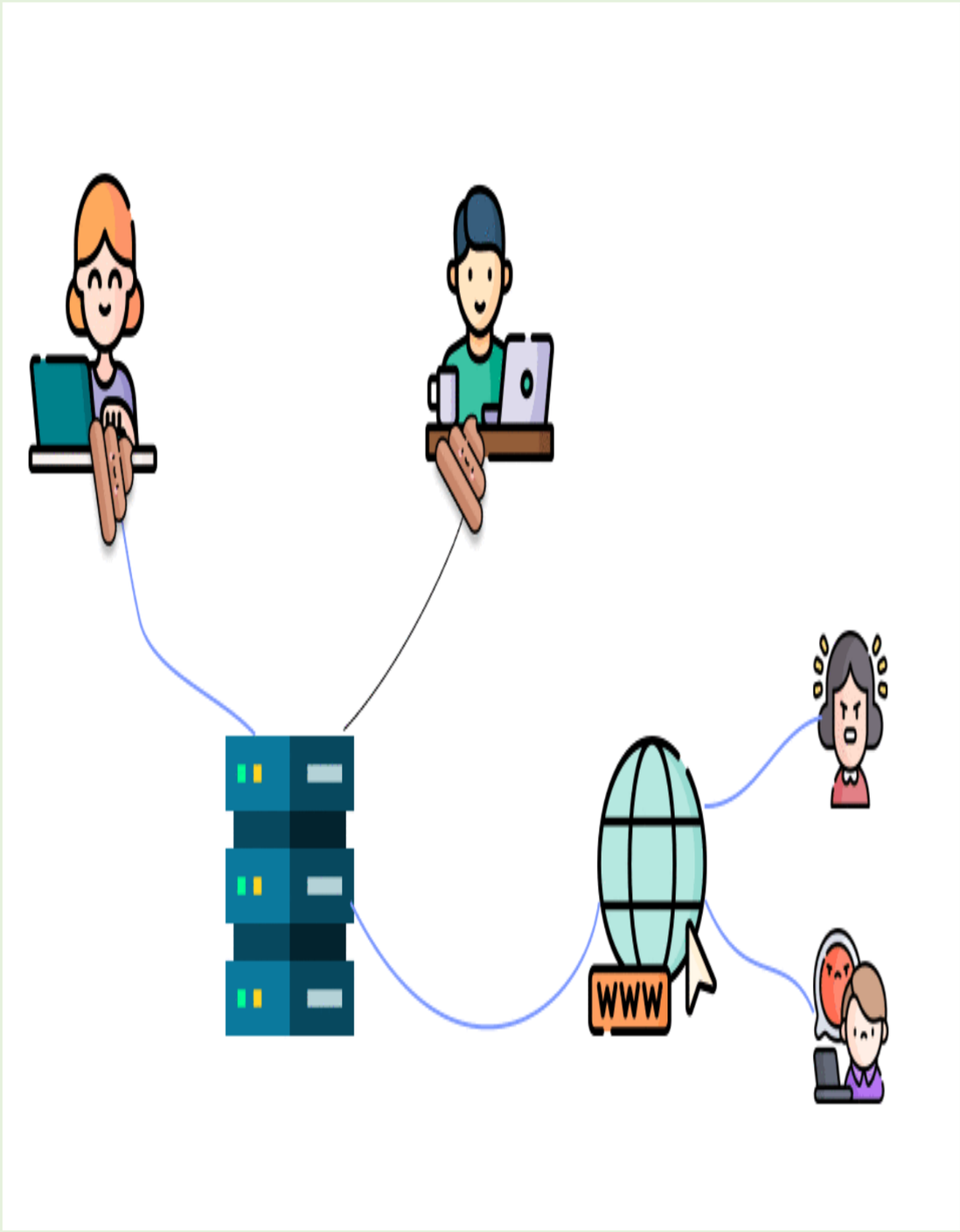
Detect unusual patterns in the data. Identify potential problems early.

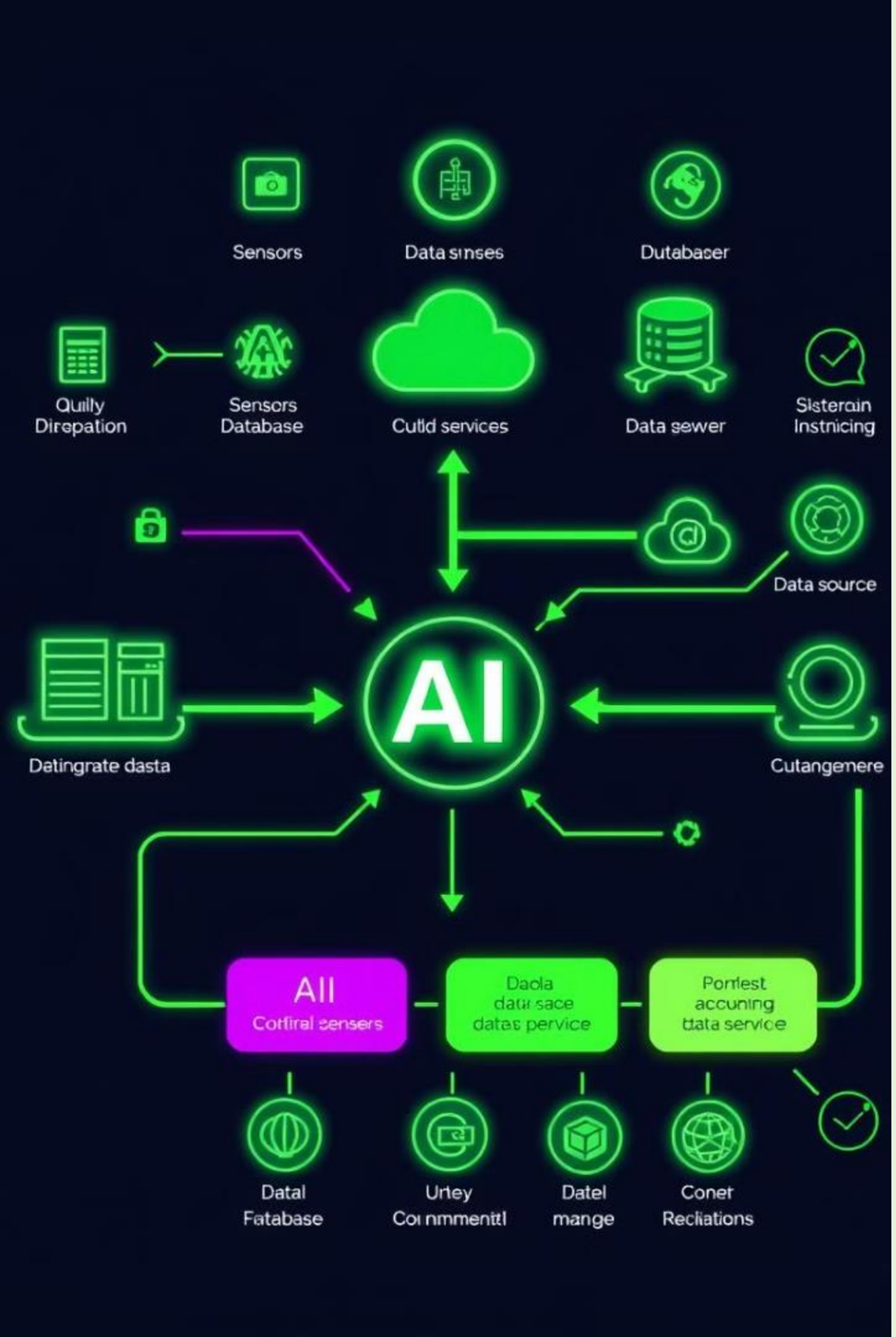
### Root Cause Analysis

Drill down to the root cause of issues. Resolve them quickly.

### Optimization Recommendations

Provide actionable tuning advice. Enhance database performance.






# Data Collection and Feature Engineering for AI-Driven Insights

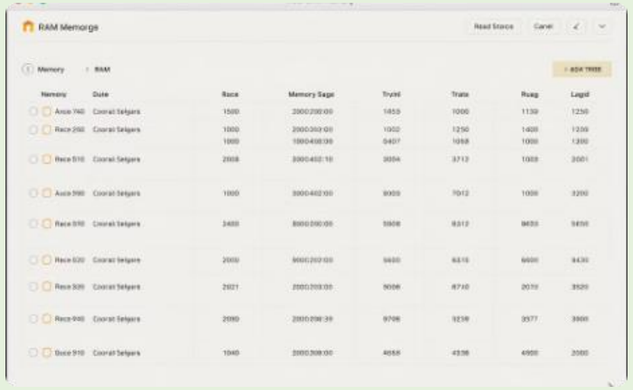
1. Log Data
2. Metrics
3. Query Data
4. System Stats

**Collect comprehensive data from PostgreSQL. Include logs, metrics, system stats, and query data. Engineer key features for AI analysis. Improve anomaly detection and insights.**

# Anomaly Detection with AI: Identifying the Root Cause

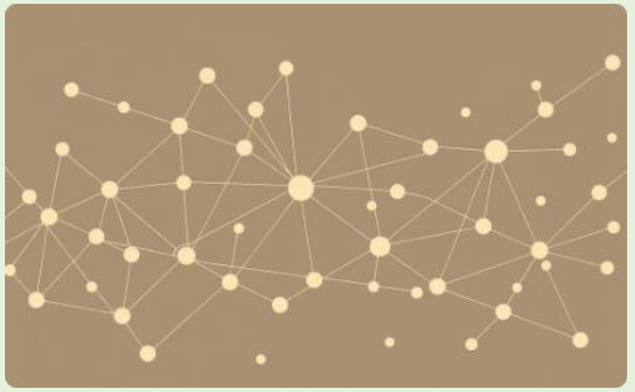
## CPU Usage Analysis

AI algorithms continuously monitor CPU utilization patterns across database operations, analyzing query execution times, background processes, and system load to detect abnormal spikes or sustained high usage. This helps identify resource-intensive queries and concurrent connection issues that may impact performance.




## Memory Usage Monitoring

Advanced machine learning models track memory consumption patterns in real-time, monitoring shared buffers, work memory allocation, and cache utilization to identify potential memory leaks, buffer overflow situations, or insufficient memory configuration.

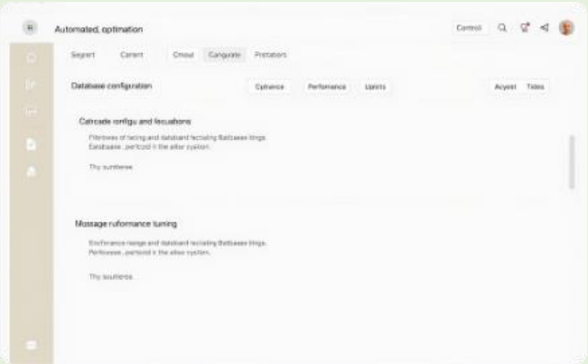


## Pattern Recognition & Root Cause Analysis

AI systems employ sophisticated pattern recognition algorithms to detect unusual behaviors, analyzing correlations between different performance indicators to quickly identify root causes. This automated approach significantly reduces traditional manual troubleshooting time.

# Predictive Optimization: Proactive Tuning Recommendations

Predictive optimization represents a paradigm shift in database management, moving from reactive troubleshooting to proactive performance tuning.



## Analyze Trends

AI algorithms analyze historical database patterns, workload characteristics, and performance metrics to establish baseline behavior.

## Predict Issues

Advanced modeling identifies potential performance bottlenecks and capacity constraints before they become critical issues.

## Recommend Tuning

System generates specific, actionable tuning recommendations to optimize database performance proactively.

By implementing these AI-driven recommendations, organizations can maintain optimal database performance, reduce operational overhead, and minimize the risk of performance-related incidents. This proactive approach leads to improved system reliability, better resource utilization, and enhanced user experience.

# Tools and Technologies for AI-Assisted Analysis

Successfully implementing AI-assisted database analysis requires a comprehensive toolkit that combines traditional PostgreSQL monitoring tools with modern AI/ML frameworks, integrated through a scalable data pipeline.


## Monitoring & Data Collection

pg_stat_statements tracks query performance with execution time and buffer hits. pg_stat_monitor provides detailed query plans, while Prometheus with custom exporters captures 20+ key metrics including buffer cache ratio and WAL generation rate.

## AI/ML Frameworks

TensorFlow powers deep learning-based anomaly detection using LSTM networks, while PyTorch implements custom attention mechanisms. Scikit-learn handles automated feature selection and dimensionality reduction.

## Visualization Tools

Grafana dashboards display customized panels for CPU, memory, and I/O metrics with alert thresholds. Jupyter notebooks with Plotly and Bokeh enable interactive exploration of performance patterns.

## Integration Platforms

Apache Kafka processes 100K+ events per second for real-time metric streaming, with custom PostgreSQL log connectors. pgBadger generates automated daily reports analyzing slow queries and connection patterns.

These integrated tools form a comprehensive performance analysis pipeline, processing over 50 different metrics in real-time to provide actionable insights for database optimization. The system can handle databases ranging from gigabytes to petabytes in size, with minimal overhead on production systems.

# AI-Powered PostgreSQL Performance Boost: An E-Commerce Success Story

**A high-traffic e-commerce platform experienced significant slowdowns in query performance, particularly during peak hours. This was causing customer dissatisfaction and lost revenue.**

1 **AI Analysis**

**The AI-powered analysis identified inefficient indexing strategies and suboptimal query plans. It pinpointed specific queries that were consuming the most resources and contributing to the overall performance bottleneck. The AI also revealed opportunities to optimize database configurations for the specific workload.**

2 **Optimization**

**Based on the AI's recommendations, missing indexes were created to improve query performance. Problematic queries were rewritten using more efficient SQL constructs and optimized join strategies. The optimization efforts focused on reducing the amount of data scanned and processed by each query.**

3 **Tuning**

The **work_mem** and **shared_buffers** settings were adjusted to better utilize available server resources. This tuning improved overall server efficiency and reduced disk I/O. Additional configuration parameters were fine-tuned based on the AI's analysis of the database's performance characteristics.

*As a result of these optimizations, query execution time decreased by 50%, significantly improving overall system performance. The e-commerce platform experienced faster page load times, improved transaction processing, and enhanced customer satisfaction. The AI-powered solution enabled the platform to handle peak traffic loads with greater efficiency and stability.*

# Revolutionizing Efficiency: Exploring AI-Driven Optimization in PostgreSQL


## Initial Implementation

Implementation at a major e-commerce platform initiated a paradigm shift from reactive to proactive database management through AI-driven optimization.


## System Deployment

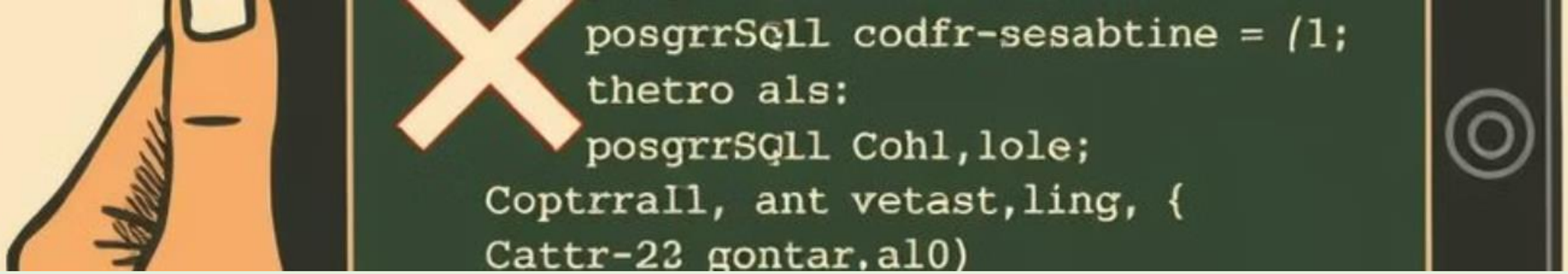
System established with automated index recommendations and dynamic resource allocation capabilities during peak loads, minimizing manual intervention requirements.


## Predictive Analytics Integration

Combined historical performance data with real-time monitoring to achieve more reliable, efficient, and scalable database operations.


## Measured Results

Achieved 40% reduction in query response times and 25% decrease in resource utilization through continuous analysis of system metrics.




# DEMO: Create and Analyze a Blocking Transaction

Follow these steps to create a blocking transaction and analyze it using SQLAI.ai:

## Step 1: Create a Blocking Transaction

Initiate a transaction in the first PostgreSQL session and acquire an advisory lock to simulate a long-running operation. This transaction will be kept open to hold the lock.

Step 2: Create a Blocked Transaction

In a second PostgreSQL session, attempt to acquire the same advisory lock, which will cause this transaction to block. This transaction will wait until the lock is released in the first session.

Step 3: Observe the Blocking

Use the `pg_stat_activity` view to observe the blocking. This view provides real-time information about PostgreSQL server activity, showing which queries are waiting and which are holding locks.

This setup will help you create and observe blocking processes in PostgreSQL.

## Analyze with SQLAI.ai:

1. Go to Generate SQL Query to check for database blocking.
2. Check for database blocking using the generated query.
3. Run the generated query in your database.
4. Identify the query causing the issue using the `pid` and `query_duration`.
5. Use "Explain Query" to understand the query execution plan.
6. Optimize the query using the "Optimize SQL Query" option to reduce blocking.

# Generate SQL Query

Turn everyday language into queries. Quick guide.

Instruction

any blocking in my database ?

Generate query  Settings

Data source  Schema  Rules (0)  Choose DB  Add

A few tips:

- Type / (slash) to use data source autosuggest from the instruction textarea.
- Set custom data source rules to improve AI.
- Share data source with team member.

```sql
SELECT
    blocked.pid AS blocked_pid,
    blocked.query AS blocked_query,
    blocker.pid AS blocking_pid,
    blocker.query AS blocking_query
FROM pg_stat_activity blocked
JOIN pg_locks bl ON blocked.pid = bl.pid
JOIN pg_locks bkl ON
     bkl.locktype = bl.locktype
  AND bkl.database IS NOT DISTINCT FROM bl.database
  AND (bkl.relation = bl.relation OR (bkl.relation IS NULL AND bl.relation
  AND bkl.page IS NOT DISTINCT FROM bl.page
  AND bkl.tuple IS NOT DISTINCT FROM bl.tuple
  AND bkl.virtualtransaction <> bl.virtualtransaction
JOIN pg_stat_activity blocker ON bkl.pid = blocker.pid
WHERE bl.granted = false
  AND bkl.granted = true;
```

Query

```
SELECT pg_advisory_lock(1);

-- Your query
SELECT * FROM person.emailaddress;
```

Optimize query | Settings

Optimizations (click to apply) | Undo | Refresh

### 1. Remove the advisory lock call if it isn't essential for read operations.

If the SELECT query on person.emailaddress is intended only for reading data and does not require synchronization, eliminating the SELECT pg_advisory_lock(1) call avoids unnecessary locking overhead and potential contention.

### 2. Replace SELECT * with an explicit column list.

Listing only the needed columns reduces the volume of data transferred and processed, which improves performance by avoiding the overhead of retrieving unneeded fields.

### 3. Enclose the lock acquisition and data retrieval in a single explicit transaction block.

Combining both the advisory lock and the SELECT query within a BEGIN/COMMIT block ensures the lock is held only for the required duration, minimizing extra round trips and reducing lock hold time.


# Conclusion: Unlock Performance with AI

## Key Takeaways

AI-assisted analysis is a game-changer: AI provides insights that are...

## Call to Action

Explore AI tools for optimization: Research and experiment with different...

# Key Takeaways

**AI-assisted analysis is a game-changer:** AI provides insights that are difficult or impossible to obtain through manual analysis, leading to more effective optimization strategies.

**DBAs reduce manual effort:** AI automates repetitive tasks, freeing up DBAs to focus on more strategic initiatives and complex problem-solving.

**Proactive management is critical:** AI enables proactive identification and resolution of performance bottlenecks, preventing issues before they impact users.

# Call to Action

**Explore AI tools for optimization:** Research and experiment with different AI-powered tools to identify the best fit for your PostgreSQL environment and performance goals.

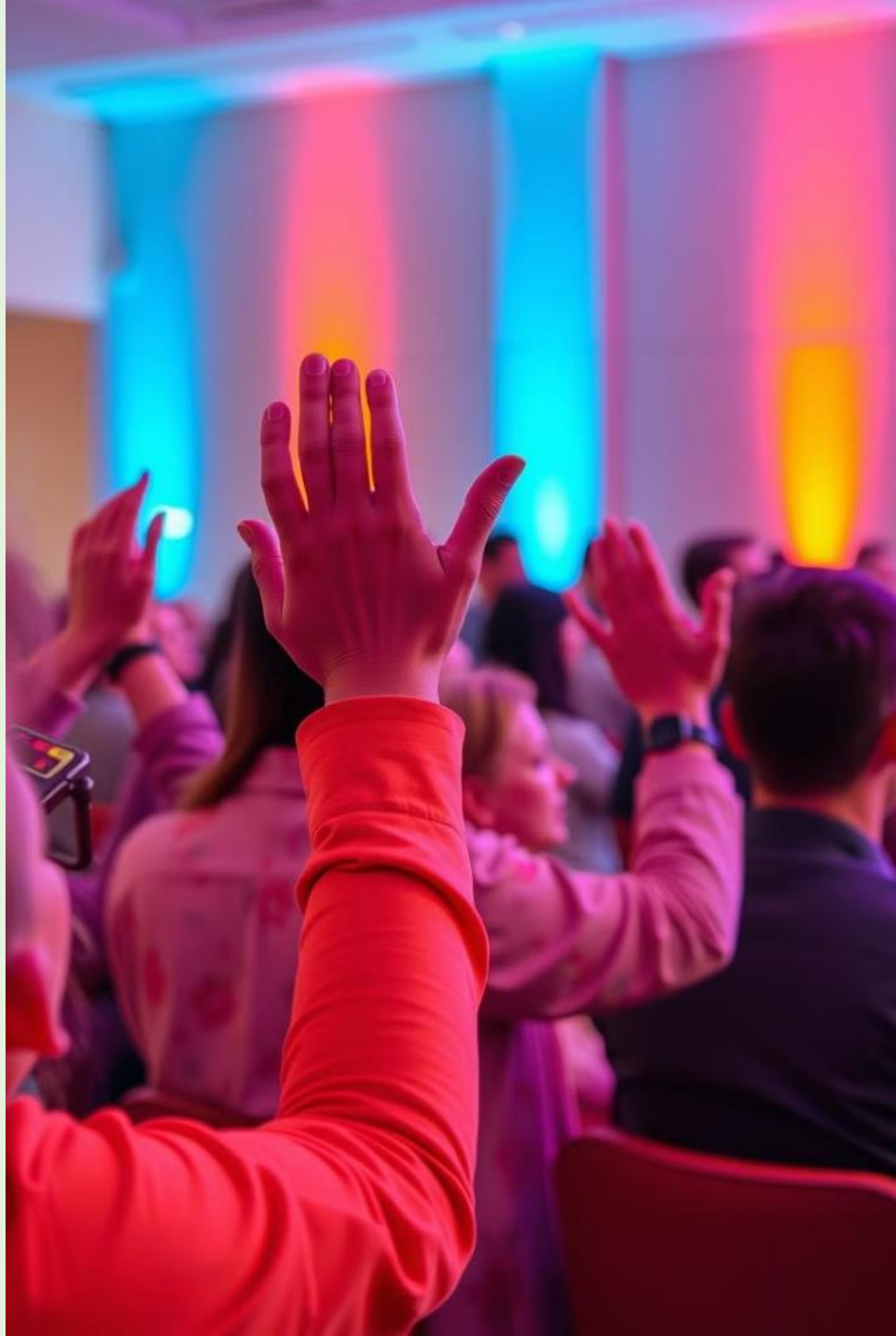
**Integrate into performance workflow:** Incorporate AI-driven insights and recommendations into your existing performance monitoring and optimization processes for continuous improvement.

# References


- https://aws.amazon.com/blogs/machine-learning/tuning-your-dbms-automatically-with-machine-learning
- **Gamma AI for PPT Deck Designs.**
- https://www.sqlai.ai/app/datasources/add/connect/postgres
- https://youtube.com/shorts/oD78F6PwmLE?si=rx2K1RnuRo1rZHyp
- **Google Images.**


# Questions and Answers

Do you have any questions? I'm here to provide clarity. Let's address your concerns. What's on your mind?



# Thank You

I Express my sincere gratitude. Your support means the world to me.

Deeply appreciate your commitment. Thank you for your time and consideration.

I  am thankful for the opportunity.

YS Let's Connect !!

LinkedIn

SQL YugData