

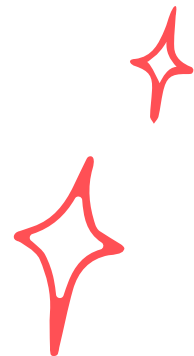
# PROJECT WALKTHROUGH AND SQL ANALYSIS

INVESTIGATING APPLICATION PERFORMANCE METRICS AND ERROR  
LOGS

An Intermediate SQL Project to Analyze Performance  
and Memory Metrics in IT Systems

Tools used : PostgreSQL , Excel , SQL

By : Nishtha Ballal

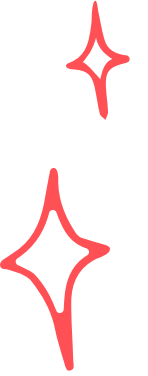


# INTRODUCTION

- The primary objective of this project is to analyze and derive insights from performance and error logs data collected from applications running in an IT environment. The goal is to identify trends, performance bottlenecks, and areas of improvement by leveraging SQL queries.
- Focus is placed on:
  - Performance metrics like response times, CPU usage, and memory usage
  - Application error logs and their relationship to performance
  - The project aims to help a Database Administrator (DBA) in managing and troubleshooting application performance.



# DATA



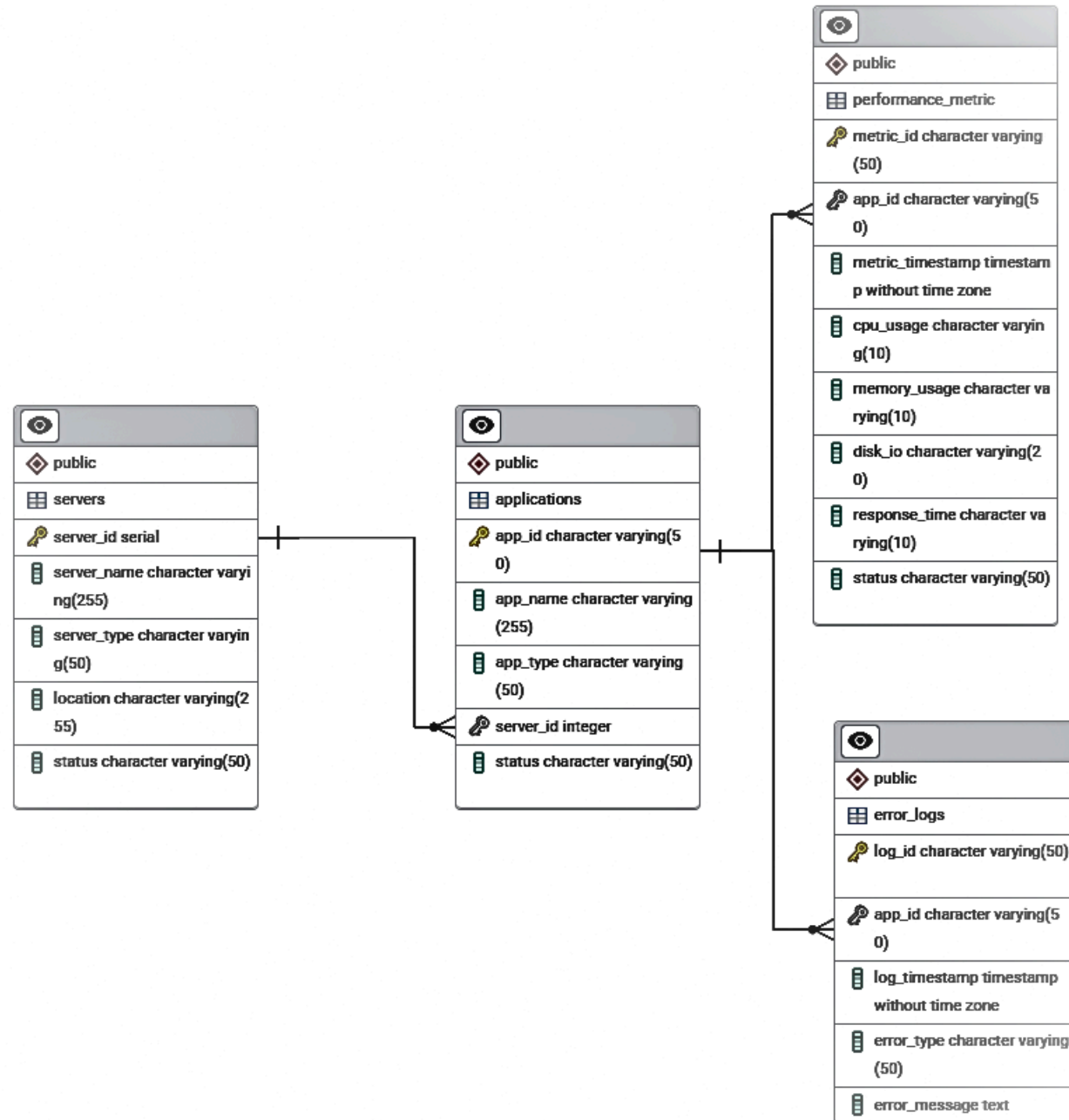
● There are four key tables :

1. **Applications Table:** Contains information about each application.
2. **Servers Table:** Contains details about the servers on which applications run.
3. **Error Log Table:** Logs errors associated with applications and their respective timestamps.
4. **Performance Metrics Table:** Contains various performance-related metrics like CPU usage, memory usage, and response times.

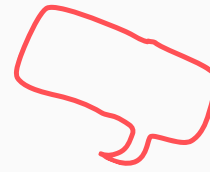
● Relationship :

1. The tables are linked through foreign keys and primary keys, ensuring data integrity.
2. Applications are associated with servers and performance metrics via `app_id` and `server_id`.

# ENTITY RELATIONSHIP DIAGRAM




# INSIGHTS



## 1. COUNT THE NUMBER OF SERVERS IN EACH LOCATION.

Outputs:

	<b>location</b> character varying (255) 🔒	<b>server_count</b> bigint 🔒
1	Datacenter2	22
2	Datacenter3	18
3	Datacenter4	13
4	Datacenter1	7



Understanding the distribution of servers across various locations is vital for analyzing infrastructure balance, redundancy, and efficiency.

Locations with a higher number of servers might indicate critical operations hubs, serving as primary data centers or regions with higher user demand.

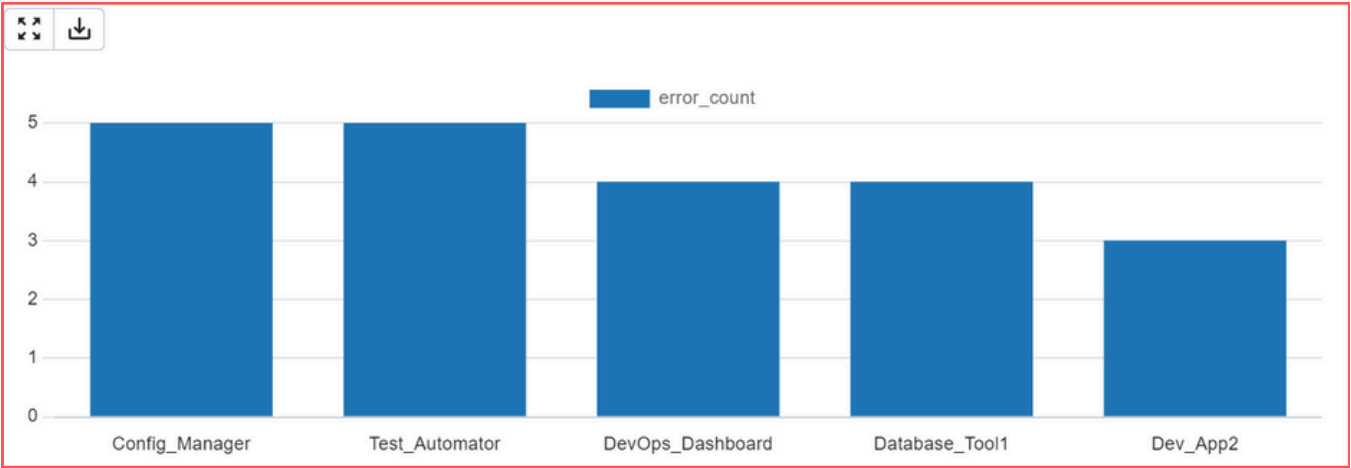
Locations with fewer servers could represent a risk if they are relied upon heavily.

## 2. IDENTIFY WHICH APPLICATION GENERATES THE MOST ERROR LOGS.

Outputs:

	app_name character varying (255) 🔒	error_count bigint 🔒
1	Config_Manager	5
2	Test_Automator	5
3	DevOps_Dashboard	4
4	Database_Tool1	4
5	Dev_App2	3


The applications **Config\_Manager** and **Test\_automator** have generated the highest number of error logs across the system.



### 3. DETERMINE THE TOP 5 APPLICATION WHOS AVERAGE RESPONSE TIME IS MORE THAN THE COLLECTIVE AVERAGE RESPONSE TIME

Outputs:

	app_name character varying (255) 🔒	avg_response_time numeric 🔒
1	System_Health	495.00
2	Deployment_Tool	490.00
3	Incident_Manager	480.00
4	Integration_Tool	470.00
5	Network_Admin	410.00



The applications **System\_Health** and **Deployment\_Tool** exhibit the highest average response times, indicating potential performance bottlenecks.

System\_Health is likely handling complex monitoring tasks or encountering high data processing loads.

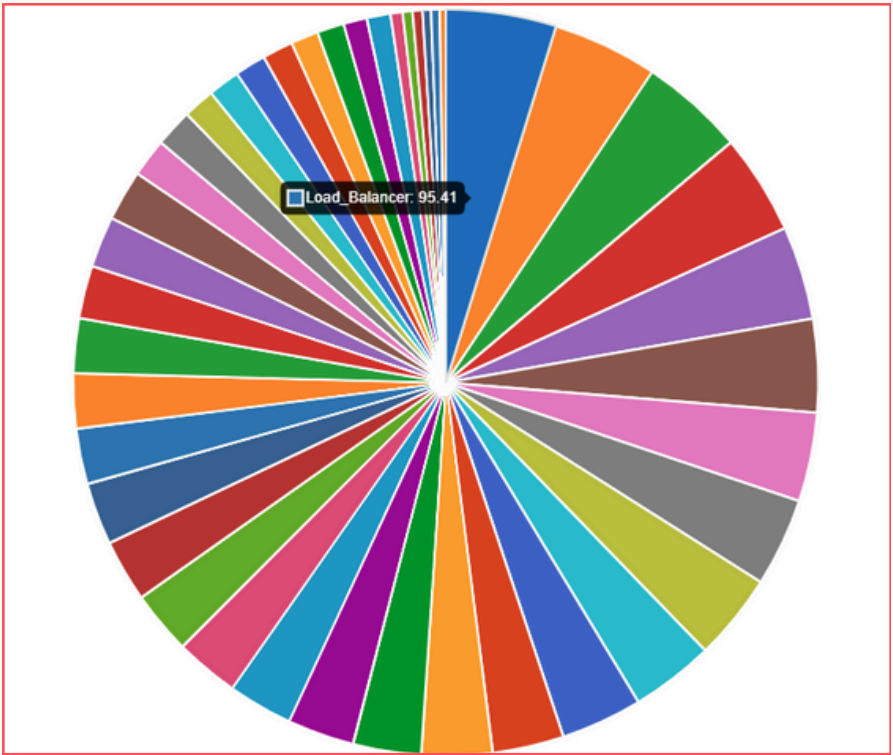
Deployment\_Tool's elevated response times suggest inefficiencies in deployment pipelines or resource contention during deployment processes.

4. TOP 3 APPLICATIONS WITH THE HIGHEST AVERAGE CPU USAGE.

Outputs:

	app_name character varying (255) 🔒	avg_cpu_usage numeric 🔒
1	Load_Balancer	95.41
2	Log_Collector	91.12
3	Workflow_Automator	90.15

The applications **Load\_Balancer** and **Log\_Collector** demonstrate the highest average CPU usage.



Average CPU usage of applications



## 5. IDENTIFY SERVERS HOSTING APPLICATIONS MARKED AS INACTIVE.

Outputs:

	server_name character varying (255) 🔒	location character varying (255) 🔒	app_name character varying (255) 🔒
1	server_echo	Datacenter3	Test_Suite
2	server_kappa	Datacenter1	Backup_Tool
3	server_psi	Datacenter4	Web_App2
4	server_omega	Datacenter4	Dev_App2
5	server_epoch	Datacenter3	Data_Analyzer
6	server_upsilon	Datacenter2	Config_Manager
7	server_zenith	Datacenter2	Security_Scanner
8	server_alpha	Datacenter3	Patch_Manager
9	server_zenon	Datacenter1	File_Sync_Tool
10	server_stratus	Datacenter3	System_Health
11	server_beta	Datacenter4	Performance_Monitor
12	server_vortex	Datacenter4	Incident_Manager
13	server_alpha	Datacenter3	Version_Control
14	server_zeta	Datacenter1	Build_Agent
15	server_omicron	Datacenter2	App_Installer
16	server_vanguard	Datacenter3	Log_Collector
17	server_infinity	Datacenter2	Resource_Tracker
18	server_spectrum	Datacenter4	Mobile_App1
19	server_spectrum	Datacenter2	Mobile_App2
20	server_pulse	Datacenter4	Integration_Tool
21	server_fusion	Datacenter3	Auth_Service
22	server_epsilon	Datacenter3	Resource_Manager
23	server_fusion	Datacenter3	Debug_Service
24	server_reactor	Datacenter2	Log_Analyzer
25	server_fusion	Datacenter3	Network_Admin
26	server_upsilon	Datacenter2	Config_Sync_Tool
27	server_altitude	Datacenter2	API_Monitor
28	server_kappa	Datacenter1	Data_Visualizer
29	server_pinnacle	Datacenter2	Key_Management_Tool

**29 servers** are currently hosting inactive applications, leading to underutilized resources.

Servers hosting inactive applications can create unnecessary overhead, such as:

- **Wasted compute resources:** Servers remain powered and maintained without meaningful tasks.
- **Increased operational costs:** Maintenance, power, and storage costs are incurred without active usage.
- **Potential performance bottlenecks:** Inactive applications might consume limited system resources inadvertently.

## 6. THE NUMBER OF ERRORS BY THEIR TYPE.

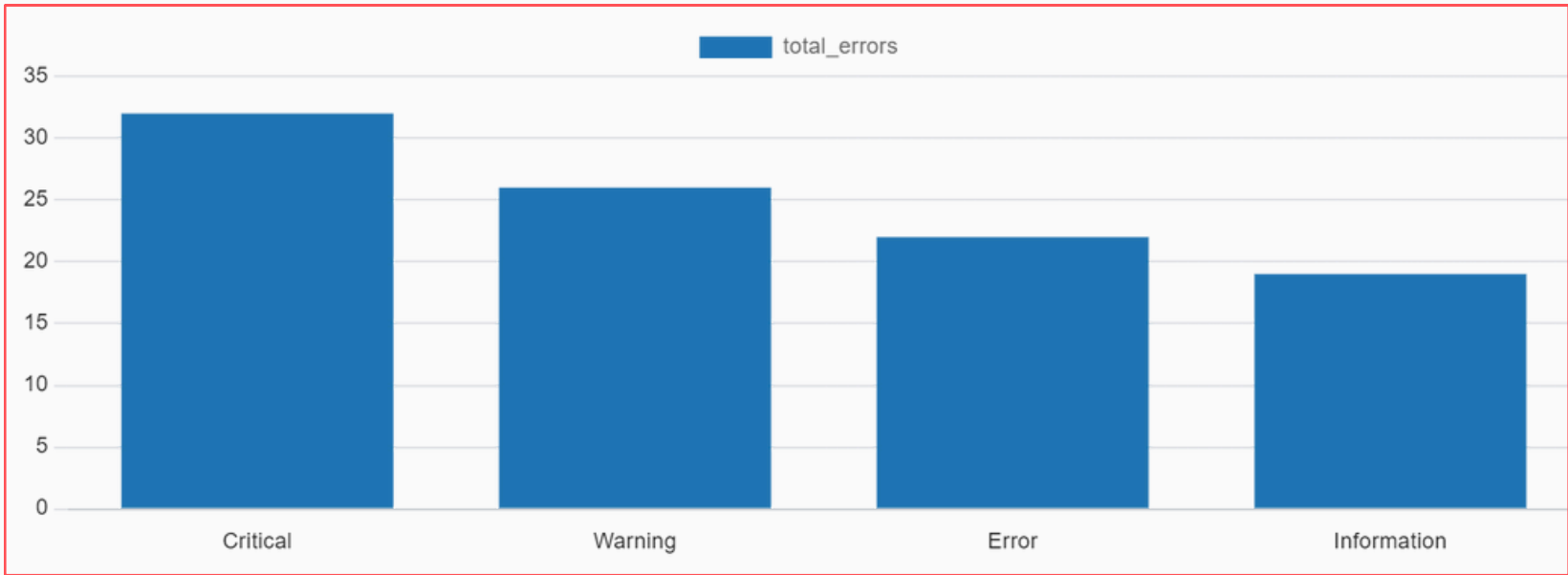
Outputs:

	error_type character varying (50) 🔒	total_errors bigint 🔒
1	Critical	32
2	Warning	26
3	Error	22
4	Information	19

**Critical** and **Warning** errors are the most frequent types of errors logged across the system.

Critical errors indicate severe issues affecting core system functionalities or applications.


While less severe than critical errors, Warning Errors still indicate potential risks or areas of concern, such as non-ideal configurations, underperformance, or minor failures that could escalate if left unresolved.



# 7. APPLICATIONS WITH MORE THAN 5 ERRORS LOGGED.

Outputs:

	<b>app_name</b> character varying (255) 🔒	<b>error_count</b> bigint 🔒
1	Config_Manager	5
2	Test_Automator	5



The applications **Config\_Manager** and **Test\_Automator** have the highest error count.

Config\_manager is likely critical for managing configurations across the infrastructure.

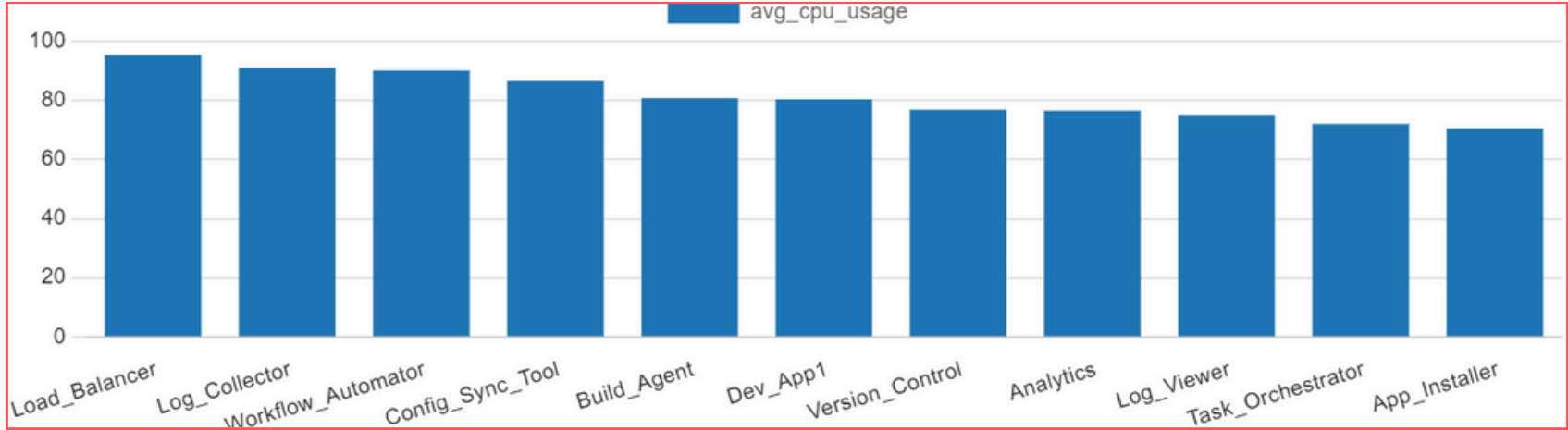
Test\_automator is likely a tool designed to automate testing, its high error rate might point to faulty test scripts or incomplete test coverage.

# 8. SERVERS WHOSE APPLICATIONS HAVE HIGH AVERAGE CPU USAGE.

Outputs:

	app_name character varying (255) 🔒	avg_cpu_usage numeric 🔒
1	Load_Balancer	95.41
2	Log_Collector	91.12
3	Workflow_Automator	90.15
4	Config_Sync_Tool	86.67
5	Build_Agent	80.85
6	Dev_App1	80.45
7	Version_Control	76.90
8	Analytics	76.57
9	Log_Viewer	75.15
10	Task_Orchestrator	72.11
11	App_Installer	70.58

The applications **Load\_Balancer** and **Log\_Collector** exhibit the highest average CPU usage, indicating that they are resource-intensive and may require optimization.



# 9. APPLICATIONS SHOWING PERFORMANCE DEGRADATION.

Applications with increasing response times over time

Outputs:

	app_name character varying (255) 🔒	response_time character varying (10) 🔒	previous_response_time character varying 🔒
1	Config_Manager	170	120
2	Workflow_Manager	220	210
3	Dev_Toolkit	300	230
4	Log_Viewer	310	150
5	App_Installer	340	200
6	Web_App1	360	120
7	Queue_Manager	410	210
8	Database_Admin	420	360
9	Debug_Service	430	100
10	Database_Admin	470	420
11	App_Installer	470	340
12	Web_App3	470	310

**12 applications** have shown a consistent increase in response times over time, indicating potential performance degradation.


These applications are likely experiencing bottlenecks due to:

- Resource constraints.
- High user demand exceeding their designed capacity.
- Code inefficiencies leading to slower processing over time.
- Dependency issues.

# 10. APPLICATIONS THAT HAVE HAD THE HIGHEST MEMORY USAGE .

Outputs:

	app_name character varying (255) 🔒	max_memory_usage text 🔒	data_points bigint 🔒
1	Auth_Service	99.72	2
2	Database_Admin	99.64	5
3	File_Sync_Tool	99.52	2
4	Web_App3	96.88	3
5	Monitoring_Tool	95.13	2
6	Debug_Service	92.85	2
7	System_Health	92.67	2
8	API_Test_Tool	87.92	2
9	Web_App2	87.38	1
10	Integration_Tool	85.71	1



The **Auth\_service** has high memory usage recorded in 2 instances. While the number of data points is low, this could indicate **isolated spikes** in memory usage, possibly during specific times of high traffic or resource-intensive operations.


The **Database\_admin** application has **5 data points** recorded with high memory usage, indicating more frequent occurrences of high memory consumption.

# 11. SERVER HOSTING THE HIGHEST NUMBER OF UNIQUE APPLICATION TYPES.

In "unique application types", we are counting the number of different app\_type values that appear for each server, ignoring duplicates.

Outputs:

	<b>server_name</b> character varying (255) 🔒	<b>unique_app_types</b> bigint 🔒
1	server_upsilon	5



**Server\_upsilon** is the server hosting the highest number of unique application types in your system.

This can say that Server\_upsilon is capable of hosting a wide range of different types of applications, suggesting that this server might be a more flexible or versatile machine in the infrastructure.

# 12. APPLICATIONS THAT HAVE NOT GENERATED ANY ERRORS.

Outputs:

	<b>app_name</b> character varying (255) 🔒	<b>app_type</b> character varying (50) 🔒
1	System_Health	Database Management System
2	Log_Analyzer	Testing Tool
3	Integration_Tool	Monitoring Tool
4	API_Monitor	Web Server
5	Cloud_Service1	Development Environment
6	Web_App1	Development Environment

These 6 applications demonstrate high stability and are likely well-configured and optimized for their intended operations.

They could serve as benchmarks for best practices in configuration, deployment, or resource allocation.

It's worth investigating if these applications are relatively low-complexity or have less user interaction compared to others. If so, their error-free state might be due to lower operational demands.

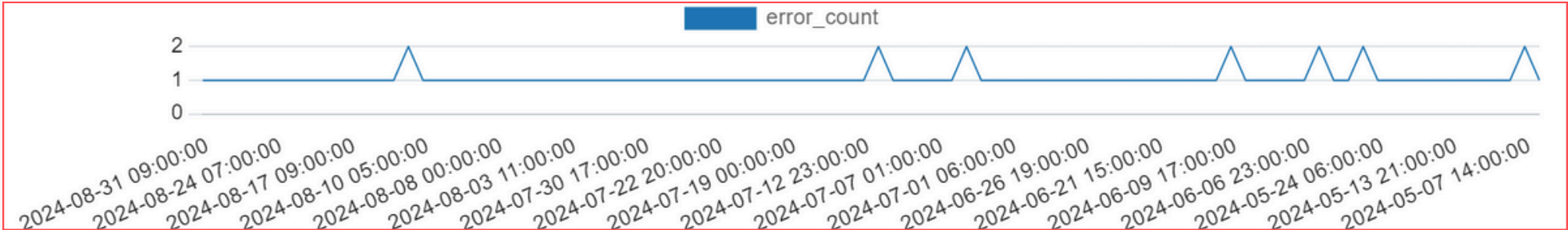


13. TIME PERIODS WITH THE HIGHEST NUMBER OF ERRORS.

Outputs:

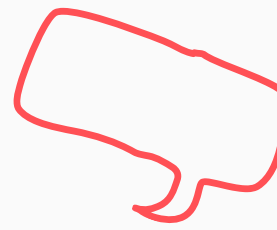
	<div>error_time<div>timestamp without time zone</div></div>	<div>error_count<div>bigint</div></div>
1	2024-06-04 22:00:00	2
2	2024-07-12 15:00:00	2
3	2024-08-12 22:00:00	2
4	2024-05-07 14:00:00	2
5	2024-06-09 17:00:00	2
6	2024-05-25 11:00:00	2
7	2024-07-04 01:00:00	2
8	2024-07-22 20:00:00	1
9	2024-08-24 07:00:00	1
10	2024-05-24 06:00:00	1

These timestamps represent the busiest or most problematic periods for the system. They may align with high user traffic, resource-intensive operations, or specific scheduled tasks. The highest number of errors occur after noon, likely tied to predictable activities such as backups, batch processing, or peak user engagement.



error count for the quater

# KEY NOTES



1. Error Generation: Config\_Manager and Test\_Automator generate the most errors, requiring urgent fixes.
2. Resource Usage:
  - CPU: High usage in Load\_Balancer and Log\_Collector indicates potential scaling needs.
  - Memory: Auth\_Service and Database\_Admin demand optimization.
3. Performance Issues:
  - System\_Health and Deployment\_Tool show highest response times.
  - 12 apps with increasing response times suggest performance degradation.
4. Inactive Apps: 29 servers host inactive apps, wasting resources.
5. Error-Free Apps: Five applications perform reliably with zero errors.
6. Trends: Peaks in errors occur during seven distinct timestamps.
7. Server Insights: Server\_Upsilon hosts the most diverse apps; server distribution shows need for improved regional redundancy.



## RECOMMENDATIONS

- Error Resolution: Focus on high-error applications and prioritize critical issues.
- Optimize Resources: Scale or optimize high CPU and memory usage apps.
- Performance Monitoring: Address bottlenecks in slow apps and monitor increasing response times.
- Inactive Servers: Decommission or repurpose servers hosting inactive apps.
- Time-Based Analysis: Investigate high-error periods for root causes.
- Server Strategy: Balance workloads regionally to ensure redundancy and growth.



THANK YOU !

