🔧 Trace Analysis & Troubleshooting

> **Series:** SPANS | **Notebook:** 3 of 8 | **Created:** December 2025

Root Cause Analysis with Distributed Traces

This notebook teaches systematic approaches to troubleshoot issues using span data. You'll learn to identify error patterns, analyze latency, and trace problems to their root cause.

Table of Contents

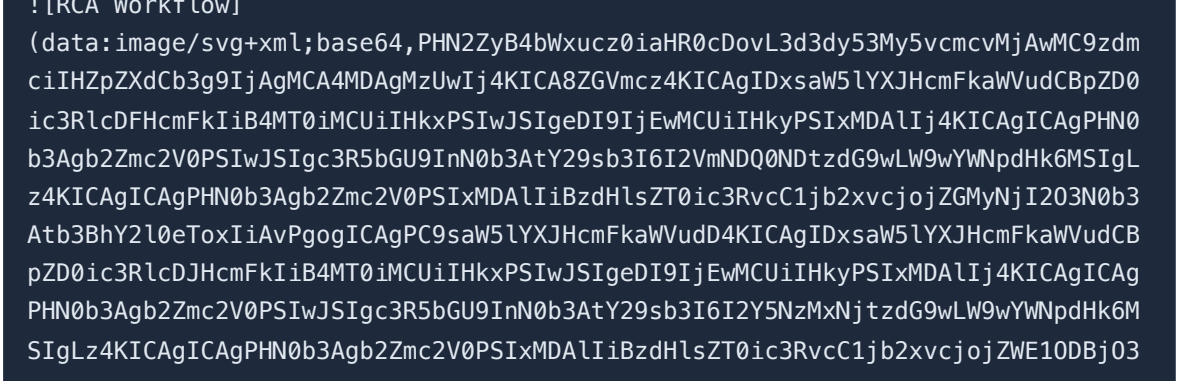
Prerequisites

Before starting this notebook, ensure you have:

- ✅ Completed **SPANS-01** and **SPANS-02**
- ✅ Access to a Dynatrace environment with span data
- ✅ Understanding of DQL filtering and aggregation

1. RCA Workflow Overview

Follow this systematic approach for root cause analysis:



N0b3Atb3BhY2l0eToxIiAvPgogICAgPC9saW5lYXJHcmFkaWVudD4KICAgIDxsaW5lYXJHcmFkaWV
udCBpZD0ic3RlcDNHcmFkIiB4MT0iMCUiIHkxPSIwJSIgeDI9IjEwMCUiIHkyPSIxMDAlIj4KICAg
ICAgPHN0b3Agb2Zmc2V0PSIwJSIgc3R5bGU9InN0b3AtY29sb3I6I2VhYjMwODtzdG9wLW9wYWNpd
Hk6MSIgLz4KICAgICAgPHN0b3Agb2Zmc2V0PSIxMDAlIiBzdHlsZT0ic3RvcC1jb2xvcjojY2E4YT
A0O3N0b3Atb3BhY2l0eToxIiAvPgogICAgPC9saW5lYXJHcmFkaWVudD4KICAgIDxsaW5lYXJHcmF
kaWVudCBpZD0ic3RlcDRHcmFkIiB4MT0iMCUiIHkxPSIwJSIgeDI9IjEwMCUiIHkyPSIxMDAlIj4K
ICAgICAgPHN0b3Agb2Zmc2V0PSIwJSIgc3R5bGU9InN0b3AtY29sb3I6IzE0OTZmZjtzdG9wLW9wY
WNpdHk6MSIgLz4KICAgICAgPHN0b3Agb2Zmc2V0PSIxMDAlIiBzdHlsZT0ic3RvcC1jb2xvcjojMG
E2NGJjO3N0b3Atb3BhY2l0eToxIiAvPgogICAgPC9saW5lYXJHcmFkaWVudD4KICAgIDxsaW5lYXJ
HcmFkaWVudCBpZD0ic3RlcDVHcmFkIiB4MT0iMCUiIHkxPSIwJSIgeDI9IjEwMCUiIHkyPSIxMDAl
Ij4KICAgICAgPHN0b3Agb2Zmc2V0PSIwJSIgc3R5bGU9InN0b3AtY29sb3I6IzIyYzU1ZTtzdG9wL
W9wYWNpdHk6MSIgLz4KICAgICAgPHN0b3Agb2Zmc2V0PSIxMDAlIiBzdHlsZT0ic3RvcC1jb2xvcj
ojMTZhMzRhO3N0b3Atb3BhY2l0eToxIiAvPgogICAgPC9saW5lYXJHcmFkaWVudD4KICAgIDxmaWx
0ZXIgaWQ9InJjYVNoYWRvdyI+CiAgICAgIDxmZURyb3BTaGFkb3cgZHg9IjIiIGR5PSIyIiBzdGRE
ZXZpYXRpb249IjMiIGZsb29kLW9wYWNpdHk9IjAuMTUiLz4KICAgIDwvZmlsdGVyPgogICAgPG1hc
mtlciBpZD0icmNhQXJyb3ciIG1hcmtlcldpZHRoPSIxMCIgbWFya2VySGVpZ2h0PSI3IiByZWZYPS
I5IiByZWZZPSIzLjUiIG9yaWVudD0iYXV0byI+CiAgICAgIDxwb2x5Z29uIHBvaW50cz0iMCAwLCA
xMCAzLjUsIDAgNyIgZmlsbD0iIzY0NzQ4YiIvPgogICAgPC9tYXJrZXI+CiAgPC9kZWZzPgoKICA8
IS0tIEJhY2tncm91bmQgLS0+CiAgPHJlY3Qgd2lkdGg9IjgwMCIgaGVpZ2h0PSIzNTAiIGZpbGw9I
iNmOGY5ZmEiIHJ4PSIxMCIvPgoKICA8IS0tIFRpdGxlIC0tPgogIDx0ZXh0IHg9IjQwMCIgeT0iMj
giIGZvbnQtZmFtaWx5PSJBcmlhbCwgc2Fucy1zZXJpZiIgZm9udC1zaXplPSIxOCIgZm9udC13ZWl
naHQ9ImJvbGQiIGZpbGw9IiMzMzMiIHRleHQtYW5jaG9yPSJtaWRkbGUiPlJvb3QgQ2F1c2UgQW5h
bHlzaXMgV29ya2Zsb3c8L3RleHQ+CgogIDwhLS0gU3RlcCAxOiBEZXRlY3QgLS0+CiAgPHJlY3Qge
D0iNDAiIHk9IjU1IiB3aWR0aD0iMTMwIiBoZWlnaHQ9IjEwMCIgcng9IjgiIGZpbGw9InVybCgjc3
RlcDFHcmFkKSIgZmlsdGVyPSJ1cmwoI3JjYVNoYWRvdykiLz4KICA8dGV4dCB4PSIxMDUiIHk9Ijg
1IiBmb250LWZhbWlseT0iQXJpYWwsIHNhbnMtc2VyaWYiIGZvbnQtc2l6ZT0iMjQiIGZpbGw9Indo
aXRlIiB0ZXh0LWFuY2hvcj0ibWlkZGxlIj4xPC90ZXh0PgogIDx0ZXh0IHg9IjEwNSIgeT0iMTEwI
iBmb250LWZhbWlseT0iQXJpYWwsIHNhbnMtc2VyaWYiIGZvbnQtc2l6ZT0iMTIiIGZvbnQtd2VpZ2
h0PSJib2xkIiBmaWxsPSJ3aGl0ZSIgdGV4dC1hbmNob3I9Im1pZGRsZSI+REVURUNUPC90ZXh0Pgo
gIDx0ZXh0IHg9IjEwNSIgeT0iMTI4IiBmb250LWZhbWlseT0iQXJpYWwsIHNhbnMtc2VyaWYiIGZv
bnQtc2l6ZT0iMTAiIGZpbGw9InJnYmEoMjU1LDI1NSwyNTUsMC45KSIgdGV4dC1hbmNob3I9Im1pZ
GRsZSI+RmluZCBlcnJvciBzcGFuczwvdGV4dD4KICA8dGV4dCB4PSIxMDUiIHk9IjE0MyIgZm9udC
1mYW1pbHk9IkFyaWFsLCBzYW5zLXNlcmlmIiBmb250LXNpemU9IjEwIiBmaWxsPSJyZ2JhKDI1NSw
yNTUsMjU1LDAuOSkiIHRleHQtYW5jaG9yPSJtaWRkbGUiPnN0YXR1c19jb2RlPWVycm9yPC90ZXh0
PgoKICA8IS0tIEFycm93IC0tPgogIDxsaW5lIHgxPSIxNzAiIHkxPSIxMDUiIHgyPSIxOTUiIHkyP
SIxMDUiIHN0cm9rZT0iIzY0NzQ4YiIgc3Ryb2tlLXdpZHRoPSIyIiBtYXJrZXItZW5kPSJ1cmwoI3
JjYUFycm93KSIvPgoKICA8IS0tIFN0ZXAgMjogUXVhbnRpZnkgLS0+CiAgPHJlY3QgeD0iMjAwIiB
5PSI1NSIgd2lkdGg9IjEzMCIgaGVpZ2h0PSIxMDAiIHJ4PSI4IiBmaWxsPSJ1cmwoI3N0ZXAyR3Jh
ZCkiIGZpbHRlcj0idXJsKCNyY2FTaGFkb3cpIi8+CiAgPHRleHQgeD0iMjY1IiB5PSI4NSIgZm9ud
C1mYW1pbHk9IkFyaWFsLCBzYW5zLXNlcmlmIiBmb250LXNpemU9IjI0IiBmaWxsPSJ3aGl0ZSIgdG
V4dC1hbmNob3I9Im1pZGRsZSI+MjwvdGV4dD4KICA8dGV4dCB4PSIyNjUiIHk9IjExMCIgZm9udC1
mYW1pbHk9IkFyaWFsLCBzYW5zLXNlcmlmIiBmb250LXNpemU9IjEyIiBmb250LXdlaWdodD0iYm9s
ZCIgZmlsbD0id2hpdGUiIHRleHQtYW5jaG9yPSJtaWRkbGUiPlFVQU5USUZZPC90ZXh0PgogIDx0Z
Xh0IHg9IjI2NSIgeT0iMTI4IiBmb250LWZhbWlseT0iQXJpYWwsIHNhbnMtc2VyaWYiIGZvbnQtc2
l6ZT0iMTAiIGZpbGw9InJnYmEoMjU1LDI1NSwyNTUsMC45KSIgdGV4dC1hbmNob3I9Im1pZGRsZSI
+RXJyb3IgcmF0ZSAmYW1wOzwvdGV4dD4KICA8dGV4dCB4PSIyNjUiIHk9IjE0MyIgZm9udC1mYW1p
bHk9IkFyaWFsLCBzYW5zLXNlcmlmIiBmb250LXNpemU9IjEwIiBmaWxsPSJyZ2JhKDI1NSwyNTUsM
jU1LDAuOSkiIHRleHQtYW5jaG9yPSJtaWRkbGUiPmltcGFjdCBhbmFseXNpczwvdGV4dD4KCiAgPC

```
EtLSBBcnJvdyAtLT4KICA8bGluZSB4MT0iMzMwIiB5MT0iMTA1IiB4Mj0iMzU1IiB5Mj0iMTA1IiB
zdHJva2U9IiM2NDc0OGIiIHN0cm9rZS13aWR0aD0iMiIgbWFya2VyLWVuZD0idXJsKCNyY2FBcnJv
dykiLz4KCiAgPCEtLSBTdGVwIDM6IExvY2F0ZSAtLT4KICA8cmVjdCB4PSIzNjAiIHk9IjU1IiB3a
WR0aD0iMTMwIiBoZWlnaHQ9IjgiIGZpbGw9InVybCgjc3RlcDNHcmFkKSIgZmlsdG
VyPSJ1cmwoI3JjYVNoYWRvdykiLz4KICA8dGV4dCB4PSI0MjUiIHk9Ijg1IiBmb250LWZhbWlseT0
iQXJpYWwsIHNhbnMtc2VyaWYiIGZvbnQtc2l6ZT0iMjQiIGZpbGw9IndoaXRlIiB0ZXh0LWFuY2hv
cj0ibWlkZGxlIj4zPC90ZXh0PgogIDx0ZXh0IHg9IjQyNSIgeT0iMTEwIiBmb250LWZhbWlseT0iQ
XJpYWwsIHNhbnMtc2VyaWYiIGZvbnQtc2l6ZT0iMTIiIGZvbnQtd2VpZ2h0PSJib2xkIiBmaWxsPS
J3aGl0ZSIgdGV4dC1hbmNob3I9Im1pZGRsZSI+TE9DQVRFPC90ZXh0PgogIDx0ZXh0IHg9IjQyNSI
geT0iMTI4IiBmb250LWZhbWlseT0iQXJpYWwsIHNhbnMtc2VyaWYiIGZvbnQtc2l6ZT0iMTAiIGZp
bGw9InJnYmEoMjU1LDI1NSwyNTUsMC45KSIgdGV4dC1hbmNob3I9Im1pZGRsZSI+V2hpY2ggc2Vyd
mljZT88L3RleHQ+CiAgPHRleHQgeD0iNDI1IiB5PSIxNDMiIGZvbnQtZmFtaWx5PSJBcmlhbCwgc2
Fucy1zZXJpZiIgZm9udC1zaXplPSIxMCIgZmlsbD0icmdiYSgyNTUsMjU1LDI1NSwwLjkpIiB0ZXh
0LWFuY2hvcj0ibWlkZGxlIj5XaGljaCBlbmRwb2ludD88L3RleHQ+CogIDwhLS0gQXJyb3cgLS0+
CiAgPGxpbmUgeDE9IjUxNSIgeTI9IjExNSIgc3Ryb2tlPSIjNjQ3NDhiIiBzdHJva2U9IndoaXRl
IiBzdHJva2Utd2lkdGg9IjIiIG1hcmtlci1lbmQ9InVybCgjcmNhQXJyb3cpIi8+CogIDwhLS0g
U3RlcCA0OiBUcmFjZSAtLT4KICA8cmVjdCB4PSI1MjAiIHk9IjU1IiB3aWR0aD0iMTMwIiBoZWl
naHQ9IjEwMCIgcng9IjIgIGZpbGw9InJnYmEoMjU1LDI1NSwyNTUsMC4wNSkiIHN0cm9rZT0iIG
YWRvdykiLz4KICA8dGV4dCB4PSI1ODUiIHk9Ijg1IiBmb250LWZhbWlseT0iQXJpYWwsIHNhbnMtc
2VyaWYiIGZvbnQtc2l6ZT0iMjQiIGZpbGw9InJnYmEoMjU1LDI1NSwyNTUsMC45KSIgdGV4dC1h
bmNob3I9Im1pZGRsZSI+VFJBQ0U8L3RleHQ+CiAgPHRleHQgeD0iNTg1IiB5PSIxMjgiIGZvbnQtZ
mFtaWx5PSJBcmlhbCwgc2Fucy1zZXJpZiIgZm9udC1zaXplPSIxMCIgZmlsbD0icmdiYSgyNTUsMj
U1LDI1NSwwLjkpIiB0ZXh0LWFuY2hvcj0ibWlkZGxlIj5GdWxsIHRyYWNlIGJ5IElEPC90ZXh0Pgo
gIDx0ZXh0IHg9IjU4NSIgeT0iMTQzIiBmb250LWZhbWlseT0iQXJpYWwsIHNhbnMtc2VyaWYiIGZv
bnQtc2l6ZT0iMTAiIGZpbGw9InJnYmEoMjU1LDI1NSwyNTUsMC45KSIgdGV4dC1hbmNob3I9Im1pZ
GRsZSI+UGFyZW50LWNoaWxkIHBhdGg8L3RleHQ+CogIDwhLS0gQXJyb3cgLS0+CiAgPGxpbmUgeD
E9IjY1MCIgeTI9IjExNSIgc3Ryb2tlPSIjNjQ3NDhiIiBzdHJva2Utd2lkdGg9IjIiIG1hcmtlc
i1lbmQ9InVybCgjcmNhQXJyb3cpIi8+CogIDwhLS0gU3RlcCA1OiBSb290IENhdXNlIC0tPgogIDx
yZWN0IHg9IjcxNSIgeT0iNTUiIHdpZHRoPSI4MCIgaGVpZ2h0PSIxMDAiIHJ4PSIyIGZpbGw9Ik
JsYWNrIiBzdHJva2U9IndoaXRlIiBzdHJva2Utd2lkdGg9IjIiIG1hcmtlci1lbmQ9InVybCgjcmN
hQXJyb3cpIi8+CiAgPHRleHQgeD0iNzU1IiB5PSIxMDUiIGZvbnQtZmFtaWx5PSJBcmlhbCwgc2Fu
cy1zZXJpZiIgZm9udC1zaXplPSIxMCIgZmlsbD0icmdiYSgyNTUsMjU1LDI1NSwwLjkpIiB0ZXh0
LWFuY2hvcj0ibWlkZGxlIj5Sb290IENhdXNlIGRlc2NyaXB0aW9uPC90ZXh0PgogIDwhLS0gU3R
lcCA2OiBTcGFuIC0tPgogIDxyZWN0IHg9IjcxNSIgeT0iMTgwIiB3aWR0aD0iODAiIGhlaWdodD
0iMTAwIiByeD0iMiIgZmlsbD0icmdiYSgyNTUsMjU1LDI1NSwwLjA1KSIgc3Ryb2tlPSJ3aGl0Z
SIgc3Ryb2tlLXdpZHRoPSIyIiBtYXJrZXItZW5kPSJ1cmwoI3JjYVNoYWRvdykiLz4KICA8dGV4dC
B4PSI3NTUiIHk9IjIwNSIgZm9udC1mYW1pbHk9IkFyaWFsLCBzYW5zLXNlcmlmIiBmb250LXNpemU
9IjEwIiBmaWxsPSJyZ2JhKDI1NSwyNTUsMjU1LDAuOSkiIHRleHQtYW5jaG9yPSJtaWRkbGUiPl
NwYW4gSUQ6IDwvdGV4dD4KICA8dGV4dCB4PSI3NTUiIHk9IjIyMCIgZm9udC1mYW1pbHk9IkFyaWF
sLCBzYW5zLXNlcmlmIiBmb250LXNpemU9IjEwIiBmaWxsPSJyZ2JhKDI1NSwyNTUsMjU1LDAuOS
kiIHRleHQtYW5jaG9yPSJtaWRkbGUiPnBhcmVudDogPC90ZXh0PgogIDx0ZXh0IHg9IjgwNSIgeT0
iMjIwIiBmb250LWZhbWlseT0iQXJpYWwsIHNhbnMtc2VyaWYiIGZvbnQtc2l6ZT0iMTAiIGZpbG
w9InJnYmEoMjU1LDI1NSwyNTUsMC45KSIgdGV4dC1hbmNob3I9InN0YXJ0Ij5zdGFydDogPC90ZXh
0PgogIDx0ZXh0IHg9IjgwNSIgeT0iMjM1IiBmb250LWZhbWlseT0iQXJpYWwsIHNhbnMtc2VyaW
YiIGZvbnQtc2l6ZT0iMTAiIGZpbGw9InJnYmEoMjU1LDI1NSwyNTUsMC45KSIgdGV4dC1hbmNob
3I9InN0YXJ0Ij5lbmQ6IDwvdGV4dD4KICA8dGV4dCB4PSI4MDUiIHk9IjI1MCIgZm9udC1mYW1p
bHk9IkFyaWFsLCBzYW5zLXNlcmlmIiBmb250LXNpemU9IjEwIiBmaWxsPSJyZ2JhKDI1NSwyNTU
sMjU1LDAuOSkiIHRleHQtYW5jaG9yPSJzdGFydCI+ZHVyYXRpb246IDwvdGV4dD4KICA8dGV4dC
B4PSI4MDUiIHk9IjI2NSIgZm9udC1mYW1pbHk9IkFyaWFsLCBzYW5zLXNlcmlmIiBmb250LXNpe
mU9IjEwIiBmaWxsPSJyZ2JhKDI1NSwyNTUsMjU1LDAuOSkiIHRleHQtYW5jaG9yPSJzdGFydCI+
c3RhdHVzX2NvZGUgPT0gImVycm9yIjwvdGV4dD4KICA8IS0tIFN0ZXAgMzogTG9jYXRlIC0tPgo
```

```
B3aWR0aD0iMjIwIiBoZWlnaHQ9IjgwIiByeD0iNiIgZmlsbD0iI2ZmZiIgc3Ryb2tlPSIjZmRiYTc
0IiBzdHJva2Utd2lkdGg9IjIiLz4KICA8dGV4dCB4PSIzOTAiIHk9IjE5NyIgZm9udC1mYW1pbHk9
IkFyaWFsLCBzYW5zLXNlcmlmIiBmb250LXNpemU9IjExIiBmb250LXdlaWdodD0iYm9sZCIgZmlsb
D0iI2Y5NzMxNiIgdGV4dC1hbmNob3I9Im1pZGRsZSI+UXVlcnk6IEXhdGVuY3kgQW5hbHlzaXM8L3
RleHQ+CiAgPHRleHQgeD0iMjk1IiB5PSIyMTgiIGZvbnQtZmFtaWx5PSJtb25vc3BhY2UiIGZvbnQ
tc2l6ZT0iMTAiIGZpbGw9IiMzMzMiPmZpbHRlciBkdXJhdGlvbiA+IDEwMDBtczwvdGV4dD4KICA8
dGV4dCB4PSIyOTUiIHk9IjIzNSIgZm9udC1mYW1pbHk9Im1vbm9zcGFjZSIgZm9udC1zaXplPSIxM
CIgZmlsbD0iIzMzMyI+fCBzdW1tYXJpemUgcDk5KGR1cmF0aW9uKTwvdGV4dD4KICAgPHJlY3QgeD
0iNTIwIiB5PSIxNzUiIHdpZHRoPSIyNDAiIGhlaWdodD0iODAiIHJ4PSI2IiBmaWxsPSIjZmZmIiB
zdHJva2U9IiM5M2M1ZmQiIHN0cm9rZS13aWR0aD0iMiIvPgogIDx0ZXh0IHg9IjY0MCIgeT0iMTk3
IiBmb250LWZhbWlseT0iQXJpYWwsIHNhbnMtc2VyaWYiIGZvbnQtc2l6ZT0iMTEiIGZvbnQtd2VpZ
2h0PSJib2xkIiBmaWxsPSIjMTQ5NmZmIiB0ZXh0LWFuY2hvcj0ibWlkZGxlIj5RdWVyeTogR2V0IE
Z1bGwgVHJhY2U8L3RleHQ+CiAgPHRleHQgeD0iNTM1IiB5PSIyMTgiIGZvbnQtZmFtaWx5PSJtb25
vc3BhY2UiIGZvbnQtc2l6ZT0iMTAiIGZpbGw9IiMzMzMiPmZpbHRlciB0cmFjZS5pZCA9PSAiYWJj
MTIzIjwvdGV4dD4KICA8dGV4dCB4PSIxMCIgeT0iMzUiIHk9IjIzNSIgZm9udC1mYW1pbHk9Im1vbm9zcGFjZ
SIgZm9udC1zaXplPSIxMCIgZmlsbD0iIzMzMyI+fCBzb3J0IHN0YXJ0X3RpbWUgYXNjPC90ZXh0Pg
oKICA8IS0tIEJvdHRvbSB0aXBzIC0tPgogIDx0ZXh0IHg9IjQwIiB5PSIyNzUiIHk9IjI3MjA
iIGhlaWdodD0iNTUiIHJ4PSI2IiBmaWxsPSIjZjBmZGY0IiBzdHJva2U9IiM4NmVmYWMiHN0cm9r
ZS13aWR0aD0iMSIvPgogIDx0ZXh0IHg9IjYwIiBmb250LWZhbWlseT0iQXJpYWwsIHNhbnMtc2VyaWYiIGZvbnQtc2l6ZT0iMTEiIGZvbnQtd2VpZ
2h0PSJib2xkIiBmaWxsPSIjMTY2NTM0Ij5UaGUgZmly
c3Qgc3BhbiBpbiB0aGUgdHJhY2Ugd2l0aCBzdGF0dXNfY29kZT0iZXJyb3IiIGlzIG9mdGVuIHRoZ
SByb290IGNhdXNlLjwvdGV4dD4KICA8dGV4dCB4PSIyMCIgeT0iMzE3IiBmb250LWZhbWlseT0iQX
JpYWwsIHNhbnMtc2VyaWYiIGZvbnQtc2l6ZT0iMTEiIGZpbGw9IiMxNjY1MzQiPlVzZSBzcGFuLnB
hcmVudF9pZCB0byB0cmFjZSB0aGUgZXJyb3IgcHJvcGFnYXRpb24gcGF0aCBmcm9tIGxlYWYgdG8g
cm9vdC48L3RleHQ+Cjwvc3ZnPgo=)



### Key Questions to Answer

| Question | Query Strategy |
|----------|----------------|
| What's failing? | Filter `span.status_code == "error"` |
| Where did it start? | Find first error in trace timeline |
| What's slow? | Filter `duration > threshold` |
| Is it widespread? | Aggregate by service/operation |
| What changed? | Compare time windows |

---

## 2. Finding Error Spans

Start by identifying error spans in your system:

> ⚠️ Remember: `span.status_code` values are lowercase (`"error"`, not
`"ERROR"`)
```

```dql
// Find recent error spans
fetch spans
| filter span.status_code == "error"
| fields start_time,
         service.name,
         span.name,
         span.status_message,
         trace.id,
         duration
| sort start_time desc
| limit 50
```

```dql
// Count errors by service to see which services are most affected
fetch spans
| filter span.status_code == "error"
| summarize {
    error_count = count(),
    affected_traces = countDistinct(trace.id)
  }, by: {service.name}
| sort error_count desc
| limit 20
```

```dql
// Error rate per service (server spans only)
fetch spans
| filter span.kind == "server"
| summarize {
    total_requests = count(),
    error_count = countIf(span.status_code == "error")
  }, by: {service.name}
| fieldsAdd error_rate_pct = (error_count * 100.0) / total_requests
| sort error_rate_pct desc
| limit 20
```

---

3. Error Pattern Analysis

Group errors to identify common patterns:

```dql
// Group errors by type and service using collectDistinct
```

```
fetch spans
| filter span.status_code == "error"
| summarize {
    occurrence = count(),
    affected_traces = countDistinct(trace.id),
    services_affected = collectDistinct(service.name)
  }, by: {span.name, span.status_message}
| sort occurrence desc
| limit 20
```

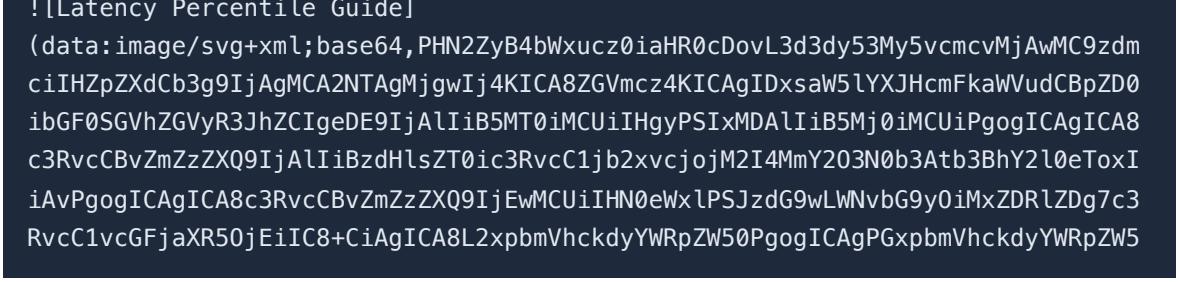
```dql
// Find traces with multiple errors (cascading failures)
fetch spans
| filter span.status_code == "error"
| summarize {
    error_count = count(),
    services_affected = collectDistinct(service.name),
    first_error = takeFirst(span.name)
  }, by: {trace.id}
| filter error_count > 1
| sort error_count desc
| limit 20
```

```dql
// Error timeline - visualize errors over time
fetch spans
| filter span.kind == "server"
| makeTimeseries {
    errors = countIf(span.status_code == "error"),
    total = count()
  }, interval: 5m, by: {service.name}
```

---

4. Latency Analysis

Analyze latency patterns to find performance issues:



```
```

0IGlkPSJsYXQ1MEdyYWQiIHgxPSIwJSIgeTE9IjAlIiB4Mj0iMTAwJSIgeTI9IjEwMCUiPgogICAg
ICA8c3RvcCBvZmZzZXQ9IjAlIiBzdHlsZT0ic3RvcC1jb2xvcjojMTBiOTgxO3N0b3Atb3BhY2l0e
ToxIiAvPgogICAgICA8c3RvcCBvZmZzZXQ9IjEwMCUiIHN0eWxlPSJzdG9wLWNvbG9yOiMwNTk2Nj
k7c3RvcC1vcGFjaXR5OjEiIC8+CiAgICA8L2xpbmVhckdyYWRpZW50PgogICAgPGxpbmVhckdyYWR
pZW50IGlkPSJsYXQ5NUdyYWQiIHgxPSIwJSIgeTE9IjAlIiB4Mj0iMTAwJSIgeTI9IjEwMCUiPgog
ICAgICA8c3RvcCBvZmZzZXQ9IjAlIiBzdHlsZT0ic3RvcC1jb2xvcjojZjU5ZTBiO3N0b3Atb3BhY
2l0eToxIiAvPgogICAgICA8c3RvcCBvZmZzZXQ9IjEwMCUiIHN0eWxlPSJzdG9wLWNvbG9yOiNkOT
c3MDY7c3RvcC1vcGFjaXR5OjEiIC8+CiAgICA8L2xpbmVhckdyYWRpZW50PgogICAgPGxpbmVhckd
yYWRpZW50IGlkPSJsYXQ5OUdyYWQiIHgxPSIwJSIgeTE9IjAlIiB4Mj0iMTAwJSIgeTI9IjEwMCUi
PgogICAgICA8c3RvcCBvZmZzZXQ9IjAlIiBzdHlsZT0ic3RvcC1jb2xvcjojZWY0NDQ0O3N0b3Atb
3BhY2l0eToxIiAvPgogICAgICA8c3RvcCBvZmZzZXQ9IjEwMCUiIHN0eWxlPSJzdG9wLWNvbG9yOi
NkYzI2MjY7c3RvcC1vcGFjaXR5OjEiIC8+CiAgICA8L2xpbmVhckdyYWRpZW50PgogICAgPGxpbmV
hckdyYWRpZW50IGlkPSJsYXRNYXhHcmFkIiB4MT0iMCUiIHkxPSIwJSIgeDI9IjEwMCUiIHkyPSIx
MDAlIj4KICAgICAgPHN0b3Agb2Zmc2V0PSIwJSIgc3R5bGU9InN0b3AtY29sb3I6IzYzNjZmMTtzd
G9wLW9wYWNpdHk6MSIgLz4KICAgICAgPHN0b3Agb2Zmc2V0PSIxMDAlIiBzdHlsZT0ic3RvcC1jb2
xvcjojNGY0NmU1O3N0b3Atb3BhY2l0eToxIiAvPgogICAgPC9saW5lYXJHcmFkaWVudD4KICAgIDx
maWx0ZXIgaWQ9ImxhdFNoYWRvdyI+CiAgICAgIDxmZURyb3BTaGFkb3cgZHg9IjIiIGR5PSIyIiBz
dGREZXZpYXRpb249IjMiIGZsb29kLW9wYWNpdHk9IjAuMTUiLz4KICAgIDwvZmlsdGVyPgogIDwvZ
GVmcz4KCiAgPCEtLSBCYWNrZ3JvdW5kIC0tPgogIDxyZWN0IHdpZHRoPSI2NTAiIGhlaWdodD0iMj
gwIiBmaWxsPSIjMGYxNzJhIiByeD0iMTAiLz4KCiAgPCEtLSBIZWFkZXIgLS0+CiAgPHJlY3QgeD0
iMzAiIHk9IjIwIiB3aWR0aD0iNTkwIiBoZWlnaHQ9IjQwIiByeD0iOCIgZmlsbD0idXJsKCNsYXRI
ZWFkZXJHcmFkKSIgZmlsdGVyPSJ1cmwoI2xhdFNoYWRvdykiLz4KICA8dGV4dCB4PSIzMjUiIHk9I
jQ3IiBmb250LWZhbWlseT0ic3lzdGVtLXVpLCAtYXBwbGUtc3lzdGVtLCBzYW5zLXNlcmlmIiBmb2
50LXNpemU9IjE2IiBmb250LXdlaWdodD0iYm9sZCIgZmlsbD0iI2ZmZiIgdGV4dC1hbmNob3I9Im1
pZGRsZSI+TGF0ZW5jeSBQZXJjZW50aWxlIEd1aWRlPC90ZXh0PgoKICA8IS0tIEZvdXIgbWV0cmlj
IGJveGVzIC0tPgogIDwhLS0gcDUwIC0tPgogIDxyZWN0IHg9IjMwIiB5PSI4MCIgd2lkdGg9IjE0M
CIgaGVpZ2h0PSI5MCIgcng9IjgiIGZpbGw9IiMxZTI5M2IiIHN0cm9rZT0iIzEwYjk4MSIgc3Ryb2
tlLXdpZHRoPSIyIiBmaWxsPSJ1cmwoI2xhdDUwR3JhZCkiLz4KICA8dGV4dCB4PSI1MCIgeT0iMTA
4IiBmb250LWZhbWlseT0ic3lzdGVtLXVpLCBzYW5zLXNlcmlmIiBmb250LXNpemU9IjE0IiBmb250
LXdlaWdodD0iYm9sZCIgZmlsbD0iI2ZmZiIgdGV4dC1hbmNob3I9Im1pZGRsZSI+cDUwPC90ZXh0P
gogIDx0ZXh0IHg9IjUwIiB5PSIxMzAiIGZvbnQtZmFtaWx5PSJzeXN0ZW0tdWksIHNhbnMtc2VyaW
YiIGZvbnQtc2l6ZT0iMTEiIGZpbGw9IiNkMWQ1ZGIiPk1lZGlhbjwvdGV4dD4KICA8dGV4dCB4PSI
1MCIgeT0iMTUwIiBmb250LWZhbWlseT0ic3lzdGVtLXVpLCBzYW5zLXNlcmlmIiBmb250LXNpemU9
IjEwIiBmaWxsPSIjOWNhM2FmIj5
UeXBpY2FsIHVzZXI8L3RleHQ+CiAgPHRleHQgeD0iNTAiIHk9IjE2MiIgZm9udC1mYW1pbHk9InN5
c3RlbS11aSwgc2Fucy1zZXJpZiIgZm9udC1zaXplPSIxMCIgZmlsbD0iIzljYTNhZiI+ZXhwZXJpZ
W5jZTwvdGV4dD4KCiAgPCEtLSBwOTUgLS0+CiAgPHJlY3QgeD0iMTg1IiB5PSI4MCIgd2lkdGg9Ij
E0MCIgaGVpZ2h0PSI5MCIgcng9IjgiIGZpbGw9IiMxZTI5M2IiIHN0cm9rZT0iI2Y1OWUwYiIgc3R
yb2tlLXdpZHRoPSIyIiBmaWxsPSJ1cmwoI2xhdDk1R3JhZCkiLz4KICA8dGV4dCB4PSIyNjUiIHk9
eT0iODAiIHdpZHRoPSIxNDAiIGhlaWdodD0iMzAiIHJ4PSI4IiBmaWxsPSJ1cmwoI2xhdDk1R3JhZ
CkiLz4KICA8cmVjdCB4PSIxODUiIHk9IjEwMCIgd2lkdGg9IjE0MCIgaGVpZ2h0PSIxMCIgZmlsbD
0idXJsKCNsYXQ5NUdyYWQpIi8+CiAgPHRleHQgeD0iMjU1IiB5PSIxMDAiIGZvbnQtZmFtaWx5PSJ
zeXN0ZW0tdWksIHNhbnMtc2VyaWYiIGZvbnQtc2l6ZT0iMTQiIGZvbnQtd2VpZ2h0PSJib2xkIiBm
aWxsPSIjZmZmIiB0ZXh0LWFuY2hvcj0ibWlkZGxlIj5wOTU8L3RleHQ+CiAgPHRleHQgeD0iMjA1I
iB5PSIxMzAiIGZvbnQtZmFtaWx5PSJzeXN0ZW0tdWksIHNhbnMtc2VyaWYiIGZvbnQtc2l6ZT0iMT
EiIGZpbGw9IiNkMWQ1ZGIiPjEgaW4gMjA8L3RleHQ+CiAgPHRleHQgeD0iMjA1IiB5PSIxNTAiIGZ

vbnQtZmFtaWx5PSJzeXN0ZW0tdWksIHNhbnMtc2VyaWYiIGZvbnQtc2l6ZT0iMTAiIGZpbGw9IiM5
Y2EzYWYiPjUlIG9mIHVzZXJzIHNlZTwvdGV4dD4KICA8dGV4dCB4PSIyMDUiIHk9IjE2MiIgZm9ud
C1mYW1pbHk9InN5c3RlbS11aSwgc2Fucy1zZXJpZiIgZm9udC1zaXplPSIxMCIgZmlsbD0iIzljYT
NhZiI+dGhpcyBvciB3b3JzZTwvdGV4dD4KCiAgPCEtLSBwOTkgLS0+CiAgPHJlY3QgeD0iMzQwIiB
5PSI4MCIgd2lkdGg9IjE0MCIgaGVpZ2h0PSI5MCIgcng9IjgiIGZpbGw9IiMxZTI5M2IiIHN0cm9r
ZT0iI2VmNDQ0NCIgc3Ryb2tlLXdpZHRoPSIyIiBmaWx0ZXI9InVybCgjbGF0U2hhZG93KSIvPgogI
DxyZWN0IHg9IjM0MCIgeT0iODAiIHdpZHRoPSIxNDAiIGhlaWdodD0iMzAiIHJ4PSI4IiBmaWxsPS
J1cmwoI2xhdDk5R3JhZCkiLz4KICA8cmVjdCB4PSIzNDAiIHk9IjEwMCIgd2lkdGg9IjE0MCIgaGV
pZ2h0PSIxMCIgZmlsbD0idXJsKCNsYXQ5OUdyYWQpIi8+CiAgPHRleHQgeD0iNDEwIiB5PSIxMDAi
IGZvbnQtZmFtaWx5PSJzeXN0ZW0tdWksIHNhbnMtc2VyaWYiIGZvbnQtc2l6ZT0iMTQiIGZvbnQtd
2VpZ2h0PSJib2xkIiBmaWxsPSIjZmZmIiB0ZXh0LWFuY2hvcj0ibWlkZGxlIj5wOTk8L3RleHQ+Ci
AgPHRleHQgeD0iMzYwIiB5PSIxMzAiIGZvbnQtZmFtaWx5PSJzeXN0ZW0tdWksIHNhbnMtc2VyaWY
iIGZvbnQtc2l6ZT0iMTEiIGZpbGw9IiNkMWQ1ZGIiPjEgaW4gMTAwPC90ZXh0PgogIDx0ZXh0IHg9
IjM2MCIgeT0iMTUwIiBmb250LWZhbWlseT0ic3lzdGVtLXVpLCBzYW5zLXNlcmlmIiBmb250LXNpe
mU9IjEwIiBmaWxsPSIjOWNhM2FmIj4xJSBvZiB1c2VycyBzZWU8L3RleHQ+CiAgPHRleHQgeD0iMz
YwIiB5PSIxNjIiIGZvbnQtZmFtaWx5PSJzeXN0ZW0tdWksIHNhbnMtc2VyaWYiIGZvbnQtc2l6ZT0
iMTAiIGZpbGw9IiM5Y2EzYWYiPnRoaXMgb3Igd29yc2U8L3RleHQ+CgogIDwhLS0gbWF4IC0tPgog
IDxyZWN0IHg9IjQ5NSIgeT0iODAiIHdpZHRoPSIxNDAiIGhlaWdodD0iOTAiIHJ4PSI4IiBmaWxsP
SIjMWUyOTNiIiBzdHJva2U9IiM2MzY2ZjEiIHN0cm9rZS13aWR0aD0iMiIgZmlsdGVyPSJ1cmwoI2
xhdFNoYWRvdykiLz4KICA8cmVjdCB4PSI0OTUiIHk9IjgwIiB3aWR0aD0iMTQwIiBoZWlnaHQ9IjM
wIiByeD0iOCIgZmlsbD0idXJsKCNsYXRNYXhHcmFkKSIvPgogIDxyZWN0IHg9IjQ5NSIgeT0iMTAw
IiB3aWR0aD0iMTQwIiBoZWlnaHQ9IjEwIiBmaWxsPSJ1cmwoI2xhdE1heEdyYWQpIi8+CiAgPHRle
HQgeD0iNTY1IiB5PSIxMDAiIGZvbnQtZmFtaWx5PSJzeXN0ZW0tdWksIHNhbnMtc2VyaWYiIGZvbn
Qtc2l6ZT0iMTQiIGZvbnQtd2VpZ2h0PSJib2xkIiBmaWxsPSIjZmZmIiB0ZXh0LWFuY2hvcj0ibWl
kZGxlIj5tYXg8L3RleHQ+CiAgPHRleHQgeD0iNTE1IiB5PSIxMzAiIGZvbnQtZmFtaWx5PSJzeXN0
ZW0tdWksIHNhbnMtc2VyaWYiIGZvbnQtc2l6ZT0iMTEiIGZpbGw9IiNkMWQ1ZGIiPldvcnN0IGNhc
2U8L3RleHQ+CiAgPHRleHQgeD0iNTE1IiB5PSIxNTAiIGZvbnQtZmFtaWx5PSJzeXN0ZW0tdWksIH
NhbnMtc2VyaWYiIGZvbnQtc2l6ZT0iMTAiIGZpbGw9IiM5Y2EzYWYiPk1heSBpbmNsdWRlPC90ZXh
0PgogIDx0ZXh0IHg9IjUxNSIgeT0iMTYyIiBmb250LWZhbWlseT0ic3lzdGVtLXVpLCBzYW5zLXNl
cmlmIiBmb250LXNpemU9IjEwIiBmaWxsPSIjOWNhM2FmIj5vdXRsaWVyczwvdGV4dD4KCiAgPCEtL
SBVc2FnZSB0aXAgLS0+CiAgPHJlY3QgeD0iMzAiIHk9IjE5MCIgd2lkdGg9IjU5MCIgaGVpZ2h0PS
I3NSIgcng9IjgiIGZpbGw9IiMxZTI5M2IiLz4KICA8dGV4dCB4PSI1MCIgeT0iMjE1IiBmb250LWZ
hbWlseT0ic3lzdGVtLXVpLCBzYW5zLXNlcmlmIiBmb250LXNpemU9IjEyIiBmb250LXdlaWdodD0i
Ym9sZCIgZmlsbD0iI2YxZjVmOSI+RFFMIEV4YW1wbGU6PC90ZXh0PgogIDx0ZXh0IHg9IjUwIiB5P
SIyNDAiIGZvbnQtZmFtaWx5PSJtb25vc3BhY2UiIGZvbnQtc2l6ZT0iMTEiIGZpbGw9IiNhNWI0Zm
MiPmZldGNoIHNwYW5zIHwgZmlsdGVyIHNwYW4ua2luZCA9PSAic2VydmVyIjwvdGV4dD4KICA8dGV
4dCB4PSI1MCIgeT0iMjU1IiBmb250LWZhbWlseT0ibW9ub3NwYWNlIiBmb250LXNpemU9IjExIiBm
aWxsPSIjYTViNGZjIj58IHN1bW1hcml6ZSBwNTA9cGVyY2VudGlsZShkdXJhdGlvbiw1MCksIHA5O
T1wZXJjZW50aWxlKGR1cmF0aW9uLDk5KSBieTp7c2VydmljZS5uYW1lfTwvdGV4dD4KPC9zdmc+Cg
==)

````
```dql
// Latency percentiles by service
fetch spans
| filter span.kind == "server"
| summarize {
````
````

```
    requests = count(),
    p50_ms = percentile(duration, 50) / 1000000,
    p95_ms = percentile(duration, 95) / 1000000,
    p99_ms = percentile(duration, 99) / 1000000,
    max_ms = max(duration) / 1000000
  }, by: {service.name}
| sort p99_ms desc
| limit 20
```

```dql
// Latency percentiles by operation
fetch spans
| filter span.kind == "server"
| summarize {
    requests = count(),
    p50_ms = percentile(duration, 50) / 1000000,
    p95_ms = percentile(duration, 95) / 1000000,
    p99_ms = percentile(duration, 99) / 1000000
  }, by: {service.name, span.name}
| filter requests > 10
| sort p95_ms desc
| limit 30
```

```dql
// Latency trend over time (use bin() for percentiles since makeTimeseries
doesn't support percentile)
fetch spans
| filter span.kind == "server"
| fieldsAdd time_bucket = bin(start_time, 10m)
| summarize {
    p95_ms = percentile(duration, 95) / 1000000,
    request_count = count()
  }, by: {time_bucket, service.name}
| sort time_bucket desc, service.name
| limit 100
```

---

5. Finding Slow Requests

Identify and analyze slow requests:

```dql
// Find slow server spans (> 1 second)
fetch spans
```

```
| filter span.kind == "server"
| filter duration > 1s
| fieldsAdd duration_ms = duration / 1000000
| fields start_time,
         service.name,
         span.name,
         duration_ms,
         trace.id
| sort duration_ms desc
| limit 50
```

```dql
// Find slow traces with summary of services involved
fetch spans
| filter span.kind == "server"
| summarize {
    trace_duration_ms = max(duration) / 1000000,
    span_count = count(),
    entry_point = takeFirst(span.name),
    services = collectDistinct(service.name)
  }, by: {trace.id}
| filter trace_duration_ms > 1000
| sort trace_duration_ms desc
| limit 20
```

```dql
// Identify slow operations (candidates for optimization)
fetch spans
| filter span.kind == "server"
| filter duration > 500ms
| summarize {
    slow_count = count(),
    avg_duration_ms = avg(duration) / 1000000,
    max_duration_ms = max(duration) / 1000000
  }, by: {service.name, span.name}
| sort slow_count desc
| limit 20
```

---

6. Reconstructing Complete Traces

Once you've identified a problematic trace, reconstruct the full picture:

```dql
```

```
// Get a sample trace ID from error spans
fetch spans
| filter span.status_code == "error"
| fields trace.id, service.name, span.name
| limit 5
```

```dql
// Reconstruct full trace (replace YOUR_TRACE_ID with actual trace.id)
fetch spans
// | filter trace.id == "YOUR_TRACE_ID"
| fieldsAdd duration_ms = duration / 1000000
| fields start_time,
         service.name,
         span.name,
         span.kind,
         duration_ms,
         span.status_code,
         span.parent_id,
         span.id
| sort start_time asc
| limit 100
```

```dql
// Analyze trace complexity
fetch spans
| summarize {
    span_count = count(),
    services_involved = countDistinct(service.name),
    has_errors = countIf(span.status_code == "error") > 0,
    total_duration_ms = sum(duration) / 1000000
  }, by: {trace.id}
| sort span_count desc
| limit 20
```

---

7. Identifying Root Cause

Use these patterns to find the origin of problems:

(
ciIHZpZXdCb3g9IjAgMCA1NTAgMzIwIj4KICA8ZGVmcz4KICAgIDxsaW5lYXJHcmFkaWVudCBpZD0
icmNhSGVhZGVyR3JhZCIgeDE9IjAlIiB5MT0iMCUiHgyPSIxMDAlIiB5Mj0iMCUiPgogICAgICA8
c3RvcCBvZmZzZXQ9IjAlIiBzdHlsZT0ic3RvcC1jb2xvcjojZWY0NDQ0O3N0b3Atb3BhY2l0eToxI

iAvPgogICAgICA8c3RvcCBvZmZzZXQ9IjEwMCUiIHN0eWxlPSJzdG9wLWNvbG9yOiNkYzI2MjY7c3
RvcC1vcGFjaXR5OjEiIC8+CiAgICA8L2xpbmVhckdyYWRpZW50PgogICAgPGxpbmVhckdyYWRpZW5
0IGlkPSJyY2FJdGVtR3JhZCIgeDE9IjAlIiB5MT0iMCUiIHgyPSIxMDAlIiB5Mj0iMTAwJSI+CiAg
ICAgIDxzdG9wIG9mZnNldD0iMCUiIHN0eWxlPSJzdG9wLWNvbG9yOiMxZTI5M2I7c3RvcC1vcGFja
XR5OjEiIC8+CiAgICAgIDxzdG9wIG9mZnNldD0iMTAwJSIgc3R5bGU9InN0b3AtY29sb3I6IzBmMT
cyYTtzdG9wLW9wYWNpdHk6MSIgLz4KICAgIDwvbGluZWFyR3JhZGllbnQ+CiAgICA8ZmlsdGVyIGl
kPSJyY2FTaGFkb3ciPgogICAgICA8ZmVEcm9wU2hhZG93IGR4PSIyIiBkeT0iMiIgc3RkRGV2aWF0
aW9uPSIzIiBmbG9vZC1vcGFjaXR5PSIwLjE1Ii8+CiAgICA8L2ZpbHRlcj4KICA8L2RlZnM+CgogI
DwhLS0gQmFja2dyb3VuZCAtLT4KICA8cmVjdCB3aWR0aD0iNTUwIiBoZWlnaHQ9IjMyMCIgZmlsbD
0iIzBmMTcyYSIgcng9IjEwIi8+CgogIDwhLS0gSGVhZGVyIC0tPgogIDxyZWN0IHg9IjMwIiB5PSI
yMCIgd2lkdGg9IjQ5MCIgaGVpZ2h0PSI0NSIgcng9IjgiIGZpbGw9InVybCgjcmNhSGVhZGVyR3Jh
ZCkiIGZpbHRlcj0idXJsKCNyY2FTaGFkb3cpIi8+CiAgPHRleHQgeD0iNTAiIHk9IjQ4IiBmb250L
WZhbWlseT0ic3lzdGVtLXVpLCAtYXBwbGUtc3lzdGVtLCBzYW5zLXNlcmlmIiBmb250LXNpemU9Ij
E2IiBmb250LXdlaWdodD0iYm9sZCIgZmlsbD0iI2ZmZiI+Um9vdCBDYXVzZSBJZGVudGlmaWNhdGl
vbiBDaGVja2xpc3Q8L3RleHQ+CiAgPHRleHQgeD0iNDkwIiB5PSI0OCIgZm9udC1mYW1pbHk9InN5
c3RlbS11aSwgc2Fucy1zZXJpZiIgZm9udC1zaXplPSIxNCIgZmlsbD0icmdiYSgyNTUsMjU1LDI1N
SwwLjgpIiB0ZXh0LWFuY2hvcj0iZW5kIj5SQ0E8L3RleHQ+CgogIDwhLS0gQ2hlY2tsaXN0IGl0ZW
1zIC0tPgogIDwhLS0gSXRlbSAxIC0tPgogIDxyZWN0IHg9IjMwIiB5PSI4MCIgd2lkdGg9IjQ5MCI
gaGVpZ2h0PSIzOCIgcng9IjYiIGZpbGw9IiMxZTI5M2IiLz4KICA8cmVjdCB4PSI0NSIgeT0iOTAi
IHdpZHRoPSIxOCIgaGVpZ2h0PSIxOCIgcng9IjQiIGZpbGw9Im5vbmUiIHN0cm9rZT0iIzEwYjk4M
SIgc3Ryb2tlLXdpZHRoPSIyIi8+CiAgPHRleHQgeD0iNzUiIHk9IjEwNCIgZm9udC1mYW1pbHk9In
N5c3RlbS11aSwgc2Fucy1zZXJpZiIgZm9udC1zaXplPSIxMiIgZmlsbD0iI2YxZjVmOSI+RmluZCB
0aGUgRklSU1QgZXJyb3IgaW4gdGhlIHRyYWNlIHRpbWVsaW5lPC90ZXh0PgogIDx0ZXh0IHg9IjQ5
MCIgeT0iMTA0IiBmb250LWZhbWlseT0ibW9ub3NwYWNlIiBmb250LXNpemU9IjEwIiBmaWxsPSIjN
mVlN2I3IiB0ZXh0LWFuY2hvcj0iZW5kIj5taW4oc3RhcnRfdGltZSk8L3RleHQ+CgogIDwhLS0gSX
RlbSAyIC0tPgogIDxyZWN0IHg9IjMwIiB5PSIxMjUiIHdpZHRoPSI0OTAiIGhlaWdodD0iMzgiIHJ
4PSI2IiBmaWxsPSIjMWUyOTNiIi8+CiAgPHJlY3QgeD0iNDUiIHk9IjEzNSIgd2lkdGg9IjE4IiBo
ZWlnaHQ9IjE4IiByeD0iNCIgZmlsbD0ibm9uZSIgc3Ryb2tlPSIjMTBiOTgxIiBzdHJva2Utd2lkd
Gg9IjIiLz4KICA8dGV4dCB4PSI3NSIgeT0iMTQ5IiBmb250LWZhbWlseT0ic3lzdGVtLXVpLCBzYW
5zLXNlcmlmIiBmb250LXNpemU9IjEyIiBmaWxsPSIjZjFmNWY5Ij5DaGVjayBpZiBlcnJvciBwcm9
wYWdhdGVkIGZyb20gYSBkb3duc3RyZWFtIHNlcnZpY2U8L3RleHQ+CiAgPHRleHQgeD0iNDkwIiB5
PSIxNDkiIGZvbnQtZmFtaWx5PSJtb25vc3BhY2UiIGZvbnQtc2l6ZT0iMTAiIGZpbGw9IiM2ZWU3Y
jciIHRleHQtYW5jaG9yPSJlbmQiPnNwYW4ua2luZD09ImNsaWVudCI8L3RleHQ+CgogIDwhLS0gSX
RlbSAzIC0tPgogIDxyZWN0IHg9IjMwIiB5PSIxNzAiIHdpZHRoPSI0OTAiIGhlaWdodD0iMzgiIHJ
4PSI2IiBmaWxsPSIjMWUyOTNiIi8+CiAgPHJlY3QgeD0iNDUiIHk9IjE4MCIgd2lkdGg9IjE4IiBo
ZWlnaHQ9IjE4IiByeD0iNCIgZmlsbD0ibm9uZSIgc3Ryb2tlPSIjMTBiOTgxIiBzdHJva2Utd2lkd
Gg9IjIiLz4KICA8dGV4dCB4PSI3NSIgeT0iMTk0IiBmb250LWZhbWlseT0ic3lzdGVtLXVpLCBzYW
5zLXNlcmlmIiBmb250LXNpemU9IjEyIiBmaWxsPSIjZjFmNWY5Ij5JZGVudGlmeSB0aGUgU0xPV0V
TVCBzcGFuIGNvbnRyaWJ1dGluZyB0byBsYXRlbmN5PC90ZXh0PgogIDx0ZXh0IHg9IjQ5MCIgeT0i
MTk0IiBmb250LWZhbWlseT0ibW9ub3NwYWNlIiBmb250LXNpemU9IjEwIiBmaWxsPSIjNmVlN2I3I
iB0ZXh0LWFuY2hvcj0iZW5kIj5tYXgoZHVyYXRpb24pPC90ZXh0PgoKICA8IS0tIEl0ZW0gNCAtLT
4KICA8cmVjdCB4PSIzMCIgeT0iMjE1IiB3aWR0aD0iNDkwIiBoZWlnaHQ9IjM4IiByeD0iNiIgZml
sbD0iIzFlMjkzYiIvPgogIDxyZWN0IHg9IjQ1IiB5PSIyMjUiIHdpZHRoPSIxOCIgaGVpZ2h0PSIx
OCIgcng9IjQiIGZpbGw9Im5vbmUiIHN0cm9rZT0iIzEwYjk4MSIgc3Ryb2tlLXdpZHRoPSIyIi8+C
iAgPHRleHQgeD0iNzUiIHk9IjIzOSIgZm9udC1mYW1pbHk9InN5c3RlbS11aSwgc2Fucy1zZXJpZi
IgZm9udC1zaXplPSIxMiIgZmlsbD0iI2YxZjVmOSI+TG9vayBmb3IgZGF0YWJhc2UgcXVlcmllcyB
vciBleHRlcm5hbCBjYWxsczwvdGV4dD4KICA8dGV4dCB4PSI0OTAiIHk9IjIzOSIgZm9udC1mYW1p
bHk9Im1vbm9zcGFjZSIgZm9udC1zaXplPSIxMCIgZmlsbD0iIzZlZTdiNyIgdGV4dC1hbmNob3I9I

mVuZCI+aXNOb3ROdWxsKGRiLnN5c3RlbSk8L3RleHQ+CgogIDwhLS0gSXRlbSA1IC0tPgogIDxyZW
N0IHg9IjMwIiB5PSIyNjAiIHdpZHRoPSI0OTAiIGhlaWdodD0iMzgiIHJ4PSI2IiBmaWxsPSIjMWU
yOTNiIi8+CiAgPHJlY3QgeD0iNDUiHk9IjI3MCIgd2lkdGg9IjE4IiBoZWlnaHQ9IjE4IiByeD0i
NCIgZmlsbD0ibm9uZSIgc3Ryb2tlPSIjMTBiOTgxIiBzdHJva2Utd2lkdGg9IjIiLz4KICA8dGV4d
CB4PSI3NSIgeT0iMjg0IiBmb250LWZhbWlseT0ic3lzdGVtLXVpLCBzYW5zLXNlcmlmIiBmb250LX
NpemU9IjEyIiBmaWxsPSIjZjFmNWY5Ij5DaGVjayBmb3IgcmV0cnkgcGF0dGVybnMgKHJlcGVhdGV
kIHNpbWlsYXIgc3BhbnMpPC90ZXh0PgogIDx0ZXh0IHg9IjQ5MCIgeT0iMjg0IiBmb250LWZhbWls
eT0ibW9ub3NwYWNlIiBmb250LXNpemU9IjEwIiBmaWxsPSIjNmVlN2I3IiB0ZXh0LWFuY2hvcj0iZ
W5kIj5jb3VudCgpIGJ5OiBzcGFuLm5hbWU8L3RleHQ+Cjwvc3ZnPgo=)


```dql
// Find the FIRST error span in failing traces
fetch spans
| filter span.status_code == "error"
| summarize {
    first_error_time = min(start_time),
    first_error_service = takeFirst(service.name),
    first_error_span = takeFirst(span.name),
    error_description = takeFirst(span.status_message)
  }, by: {trace.id}
| sort first_error_time desc
| limit 20
```

```dql
// Find the bottleneck span in each trace
fetch spans
| summarize {
    max_duration_ms = max(duration) / 1000000,
    total_spans = count(),
    services = collectDistinct(service.name)
  }, by: {trace.id}
| filter max_duration_ms > 500
| sort max_duration_ms desc
| limit 20
```

```dql
// Time spent per span kind (where is time going?)
fetch spans
| summarize {
    total_time_ms = sum(duration) / 1000000,
    span_count = count(),
    avg_time_ms = avg(duration) / 1000000
  }, by: {span.kind}
| sort total_time_ms desc
```

```

---

## 8. Downstream Dependency Analysis

Analyze failures in downstream services (CLIENT spans):

```dql
// Find which downstream services are causing errors
fetch spans
| filter span.kind == "client" and span.status_code == "error"
| summarize {
    failure_count = count(),
    sample_error = takeFirst(span.status_message),
    affected_traces = countDistinct(trace.id)
  }, by: {service.name, span.name}
| sort failure_count desc
| limit 20
```

```dql
// Dependency health - error rates for outbound calls
fetch spans
| filter span.kind == "client"
| summarize {
    calls = count(),
    errors = countIf(span.status_code == "error"),
    p95_latency_ms = percentile(duration, 95) / 1000000
  }, by: {service.name, span.name}
| fieldsAdd error_rate_pct = (errors * 100.0) / calls
| filter error_rate_pct > 5 or p95_latency_ms > 500
| sort error_rate_pct desc
| limit 20
```

```dql
// Map service-to-service dependencies
fetch spans
| filter span.kind == "client"
| filter isNotNull(server.address)
| summarize {
    call_count = count(),
    error_count = countIf(span.status_code == "error"),
    avg_latency_ms = avg(duration) / 1000000
  }, by: {service.name, server.address}
| fieldsAdd error_rate_pct = (error_count * 100.0) / call_count
| sort call_count desc
```
```

````
| limit 30
```

---

## 9. Database Query Troubleshooting

Analyze database operations for performance issues:

```dql
// Find slow database queries
fetch spans
| filter isNotNull(db.system)
| filter duration > 100ms
| summarize {
    query_count = count(),
    avg_ms = avg(duration) / 1000000,
    p95_ms = percentile(duration, 95) / 1000000,
    max_ms = max(duration) / 1000000
  }, by: {db.system, db.name, span.name}
| sort p95_ms desc
| limit 20
```

```dql
// Find failing database operations
fetch spans
| filter isNotNull(db.system) and span.status_code == "error"
| summarize {
    error_count = count(),
    sample_error = takeFirst(span.status_message)
  }, by: {db.system, db.name, span.name}
| sort error_count desc
| limit 20
```

```dql
// Quick service health check (use for dashboards)
fetch spans
| filter span.kind == "server"
| summarize {
    requests = count(),
    errors = countIf(span.status_code == "error"),
    p50_ms = percentile(duration, 50) / 1000000,
    p99_ms = percentile(duration, 99) / 1000000
  }, by: {service.name}
| fieldsAdd error_rate_pct = (errors * 100.0) / requests
| sort error_rate_pct desc
````

```
| limit 20
```

Summary

In this notebook, you learned:

✅ **RCA Workflow** - Systematic approach: Detect → Isolate → Analyze → Correlate → Resolve
✅ **Find errors** using `span.status_code == "error"` and count affected traces
✅ **Error patterns** with `collectDistinct()` to see affected services
✅ **Latency analysis** using percentiles (p50, p95, p99) and `bin()` for trends
✅ **Slow request analysis** to identify optimization candidates
✅ **Trace reconstruction** to see the full picture
✅ **Root cause identification** - first error, slowest span, bottlenecks
✅ **Dependency analysis** - CLIENT spans show downstream failures
✅ **Database troubleshooting** for slow or failing queries

Next Steps

Continue to **SPANS-04: Service Dependencies & Flow Analysis** to learn:
- Mapping service-to-service relationships
- Analyzing async messaging patterns
- Visualizing request flows
- Critical path analysis