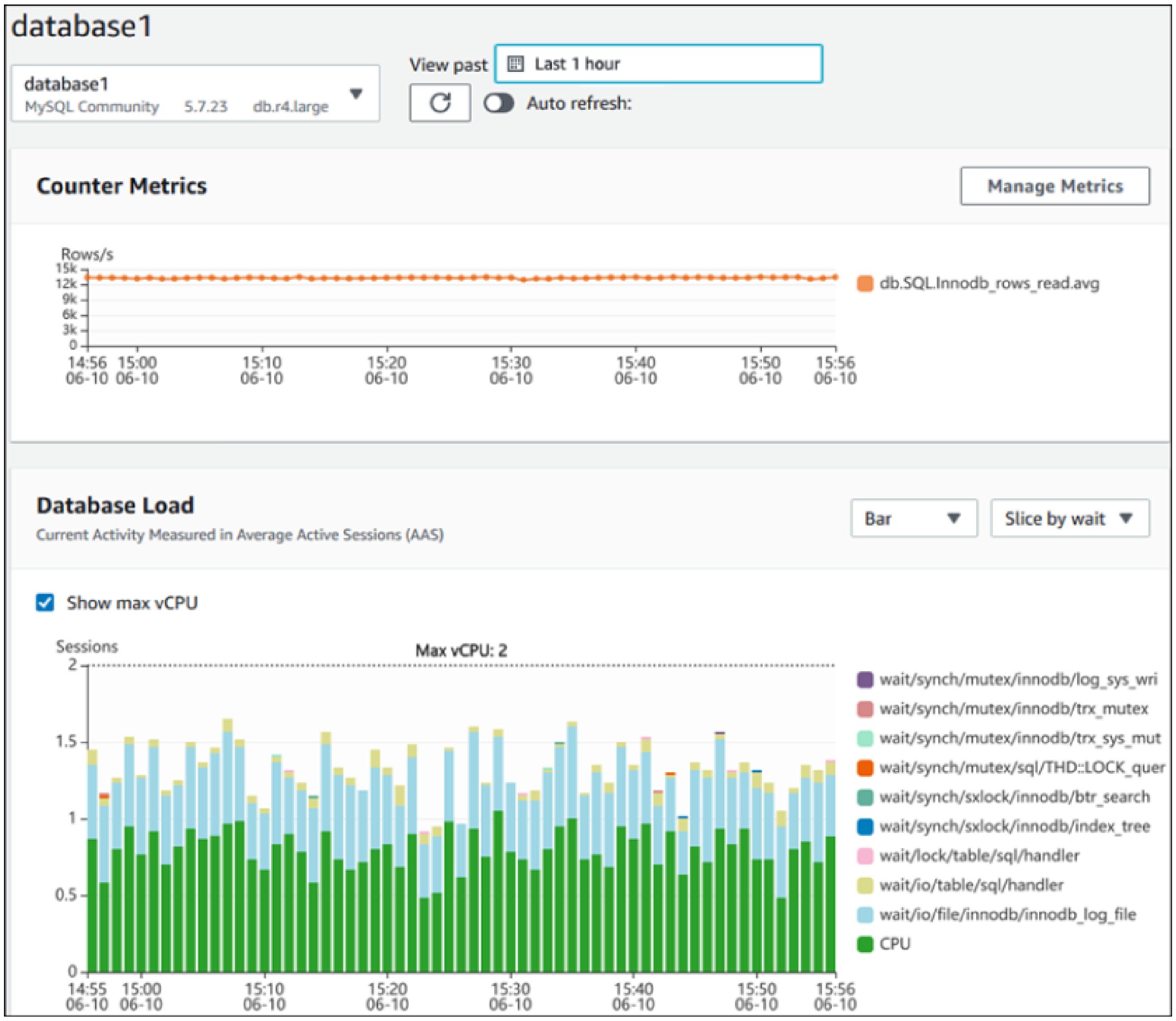
Monitoring DB load with Performance Insights on Amazon RDS

Performance Insights expands on existing Amazon RDS monitoring features to illustrate and help you analyze your database performance. With the Performance Insights dashboard, you can visualize the database load on your Amazon RDS DB instance load and filter the load by waits, SQL statements, hosts, or users.

By default, Performance Insights is turned on in the console create wizard for all Amazon RDS engines. If you have more than one database on a DB instance, Performance Insights aggregates performance data.

The dashboard is the easiest way to interact with Performance Insights. The following example shows the dashboard for a MySQL DB instance.



database1
database1
MySQL Community 5.7.23 db.r4.large
View past
Last 1 hour
Auto refresh:
Counter Metrics
Manage Metrics
Rows/s
15k
12k
9k
6k
3k
0
14:56 06-10
15:00 06-10
15:10 06-10
15:20 06-10
15:30 06-10
15:40 06-10
15:50 06-10
15:56 06-10
db.SQL.Innodb_rows_read.avg
Database Load
Current Activity Measured in Average Active Sessions (AAS)
Bar
Slice by wait
Show max vCPU
Sessions
Max vCPU: 2
2
1.5
1
0.5
0
14:55 06-10
15:00 06-10
15:10 06-10
15:20 06-10
15:30 06-10
15:40 06-10
15:50 06-10
15:56 06-10
wait/synch/mutex/innodb/log_sys_wri
wait/synch/mutex/innodb/trx_mutex
wait/synch/mutex/innodb/trx_sys_mut
wait/synch/mutex/sql/THD::LOCK_quer
wait/synch/sxlock/innodb/btr_search
wait/synch/sxlock/innodb/index_tree
wait/lock/table/sql/handler
wait/io/table/sql/handler
wait/io/file/innodb/innodb_log_file
CPU

We can view following on this tool

Database load : Database load (DB load) measures the level of session activity in your database. The key metric in Performance Insights is DBLoad, which is collected every second.

Active sessions

A *database session* represents an application's dialogue with a relational database. An *active session* is a connection that has submitted work to the DB engine and is waiting for a response.

A session is active when it's either running on CPU or waiting for a resource to become available so that it can proceed. For example, an active session might wait for a page (or block) to be read into memory, and then consume CPU while it reads data from the page.

Average active sessions

The average active sessions (AAS) is the unit for the DBLoad metric in Performance Insights. It measures how many sessions are concurrently active on the database.

Every second, Performance Insights samples the number of sessions concurrently running a query. For each active session, Performance Insights collects the following data:

- SQL statement
- Session state (running on CPU or waiting)
- Host
- User running the SQL

Performance Insights calculates the AAS by dividing the total number of sessions by the number of samples for a specific time period. For example, the following table shows 5 consecutive samples of a running query taken at 1-second intervals.

Sample	Number of sessions running query	AAS	Calculation
1	2	2	2 total sessions / 1 sample
2	0	1	2 total sessions / 2 samples
3	4	2	6 total sessions / 3 samples
4	0	1.5	6 total sessions / 4 samples
5	4	2	10 total sessions / 5 samples

In the preceding example, the DB load for the time interval was 2 AAS. This measurement means that, on average, 2 sessions were active at any given time during the interval when the 5 samples were taken.

An analogy for DB load is worker activity in a warehouse. Suppose that the warehouse employs 100 workers. If 1 order comes in, 1 worker fulfills the order while 99 workers are idle. If 100 orders come in, all 100 workers fulfill orders simultaneously. If every 15 minutes a manager writes down how many workers are simultaneously active, adds these numbers at the end of the day, and then divides the total by the number of samples, the manager calculates the average number of workers active at any given time. If the average was 50 workers yesterday and 75 workers today, then the average activity level in the warehouse increased. Similarly, DB load increases as database session activity increases.

Average active executions

The average active executions (AAE) per second is related to AAS. To calculate the AAE, Performance Insights divides the total execution time of a query by the time interval. The following table shows the AAE calculation for the same query in the preceding table.

Elapsed time (sec)	Total execution time (sec)	AAE	Calculation
60	120	2	120 execution seconds/60 elapsed seconds
120	120	1	120 execution seconds/120 elapsed seconds
180	380	2.11	380 execution seconds/180 elapsed seconds
240	380	1.58	380 execution seconds/240 elapsed seconds
300	600	2	600 execution seconds/300 elapsed seconds

In most cases, the AAS and AAE for a query are approximately the same. However, because the inputs to the calculations are different data sources, the calculations often vary slightly.

Dimensions

The db.load metric is different from the other time-series metrics because you can break it into subcomponents called dimensions. You can think of dimensions as "slice by" categories for the different characteristics of the DBLoad metric.

When you are diagnosing performance issues, the following dimensions are often the most useful:

Wait events

A wait event causes a SQL statement to wait for a specific event to happen before it can continue running. Wait events are an important dimension, or category, for DB load because they indicate where work is impeded.

Every active session is either running on the CPU or waiting. For example, sessions consume CPU when they search memory for a buffer, perform a calculation, or run procedural code. When sessions aren't consuming CPU, they might be waiting for a memory buffer to become free, a data file to be read, or a log to be written to. The more time that a session waits for resources, the less time it runs on the CPU.

When you tune a database, you often try to find out the resources that sessions are waiting for. For example, two or three wait events might account for 90 percent of DB load. This measure means that, on average, active sessions are spending most of their time waiting for a small number of resources. If you can find out the cause of these waits, you can attempt a solution.

Consider the analogy of a warehouse worker. An order comes in for a book. The worker might be delayed in fulfilling the order. For example, a different worker might be currently restocking the shelves, a trolley might not be available. Or the system used to enter the order status might be slow. The longer the worker waits, the longer it takes to fulfill the order. Waiting is a natural part of the warehouse workflow, but if wait time becomes excessive, productivity decreases. In the same way, repeated or lengthy session waits can degrade database performance.

If the Database load chart shows a bottleneck, you can find out where the load is coming from. To do so, look at the top load items table below the Database load chart. Choose a particular item, like a SQL query or a user, to drill down into that item and see details about it.

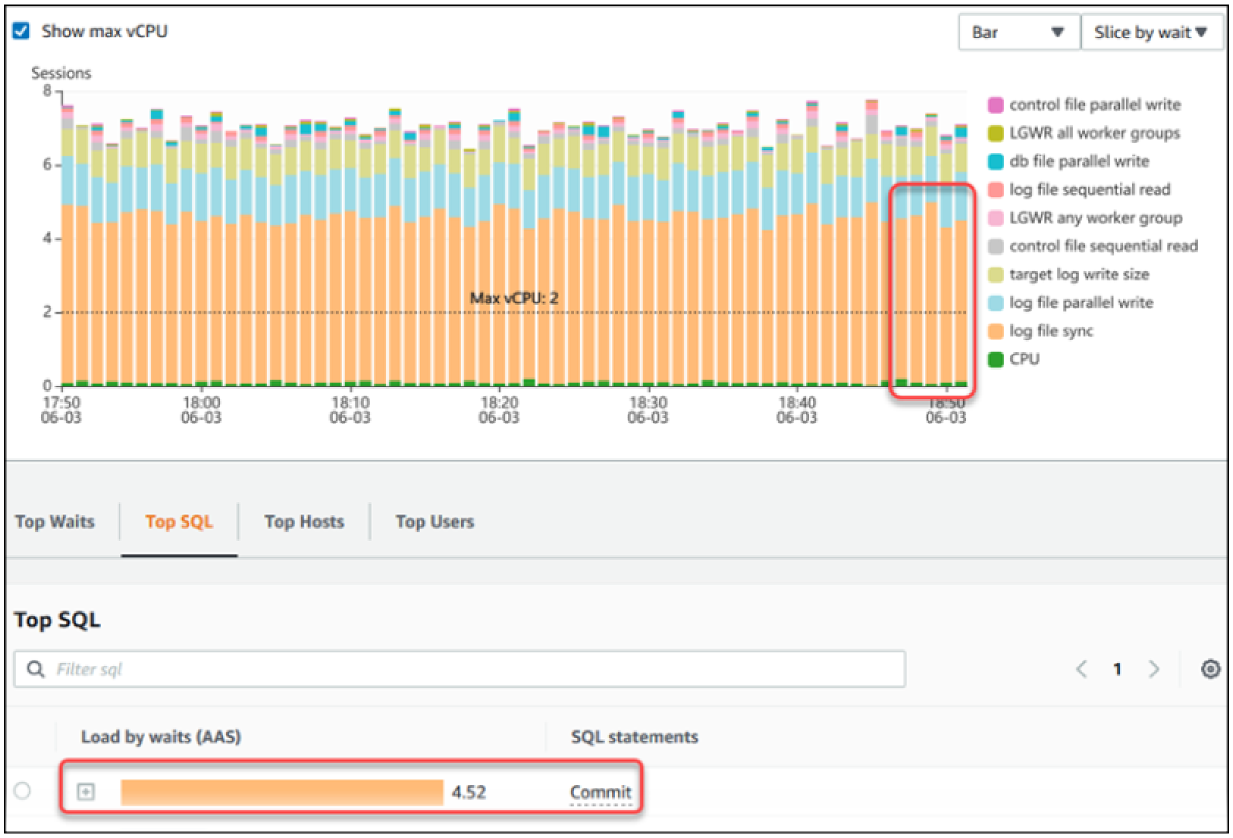
DB load grouped by waits and top SQL queries is the default Performance Insights dashboard view. This combination typically provides the most insight into performance issues. DB load grouped by waits

shows if there are any resource or concurrency bottlenecks in the database. In this case, the SQL tab of the top load items table shows which queries are driving that load.

Your typical workflow for diagnosing performance issues is as follows:

- Review the Database load chart and see if there are any incidents of database load exceeding the Max CPU line.
- If there is, look at the Database load chart and identify which wait state or states are primarily responsible.
- Identify the digest queries causing the load by seeing which of the queries the SQL tab on the top load items table are contributing most to those wait states. You can identify these by the DB Load by Wait column.
- Choose one of these digest queries in the SQL tab to expand it and see the child queries that it is composed of.
-

For example, in the dashboard following, log file sync waits account for most of the DB load. The LGWR all worker groups wait is also high. The Top SQL chart shows what is causing the log file sync waits: frequent COMMIT statements. In this case, committing less frequently will reduce DB load.



Top SQL

Load by waits (AAS)	SQL statements
4.52	Commit

Top SQL

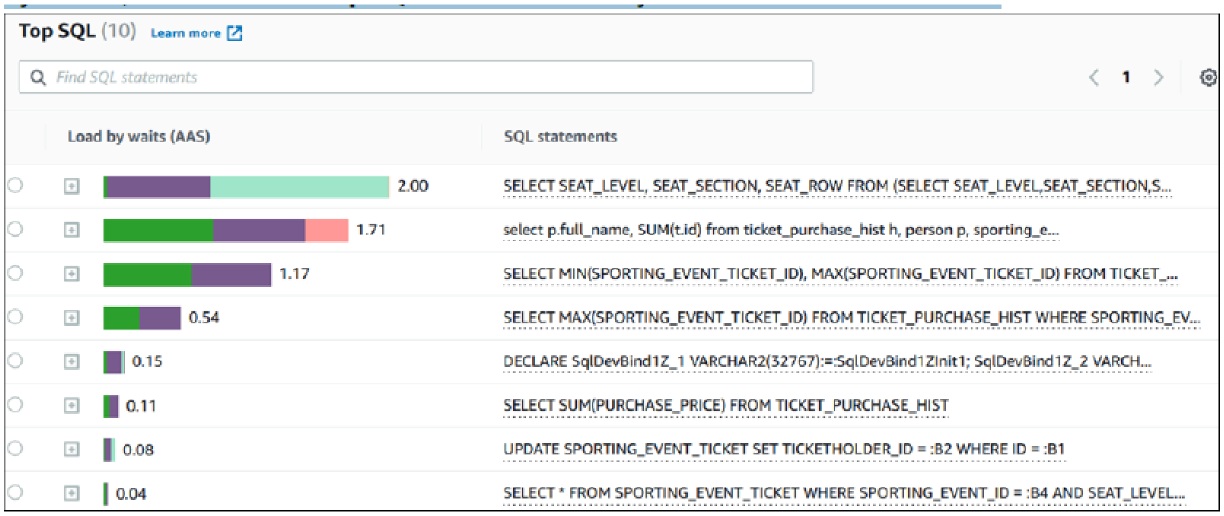
Where wait events show bottlenecks, top SQL shows which queries are contributing the most to DB load. For example, many queries might be currently running on the database, but a single query might consume 99 percent of the DB load. In this case, the high load might indicate a problem with the query.

By default, the Performance Insights console displays top SQL queries that are contributing to the database load. The console also shows relevant statistics for each statement. To diagnose performance problems for a specific statement, you can examine its execution plan.

By default, the Top SQL tab shows the 25 queries that are contributing the most to DB load. To help tune your queries, you can analyze information such as the query text and SQL statistics. You can also choose the statistics that you want to appear in the Top SQL tab.

SQL text

By default, each row in the Top SQL table shows 500 bytes of text for each statement.



Top SQL (10) Learn more

Load by waits (AAS)	SQL statements
2.00	SELECT SEAT_LEVEL, SEAT_SECTION, SEAT_ROW FROM (SELECT SEAT_LEVEL,SEAT_SECTION,S...
1.71	select p.full_name, SUM(t.id) from ticket_purchase_hist h, person p, sporting_e...
1.17	SELECT MIN(SPORTING_EVENT_TICKET_ID), MAX(SPORTING_EVENT_TICKET_ID) FROM TICKET_...
0.54	SELECT MAX(SPORTING_EVENT_TICKET_ID) FROM TICKET_PURCHASE_HIST WHERE SPORTING_EV...
0.15	DECLARE SqlDevBind1Z_1 VARCHAR2(32767):=:SqlDevBind1ZInit1; SqlDevBind1Z_2 VARCH...
0.11	SELECT SUM(PURCHASE_PRICE) FROM TICKET_PURCHASE_HIST
0.08	UPDATE SPORTING_EVENT_TICKET SET TICKETHOLDER_ID = :B2 WHERE ID = :B1
0.04	SELECT * FROM SPORTING_EVENT_TICKET WHERE SPORTING_EVENT_ID = :B4 AND SEAT_LEVEL...

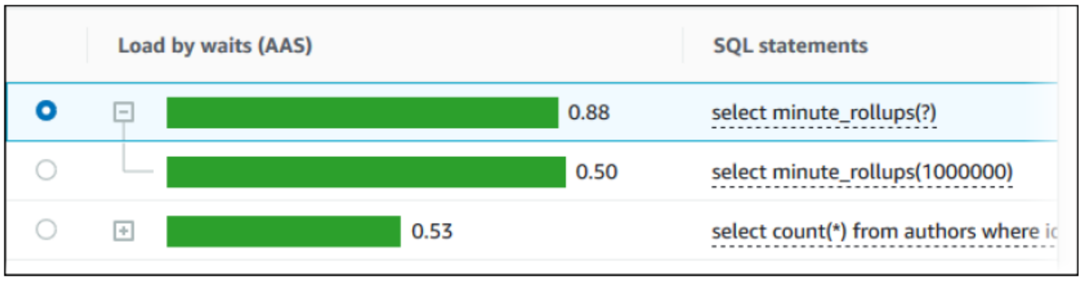
A SQL digest is a composite of multiple actual queries that are structurally similar but might have different literal values. The digest replaces hardcoded values with a question mark. For example, a digest might be SELECT * FROM emp WHERE lname= ?. This digest might include the following child queries:

SELECT * FROM emp WHERE lname = 'Sanchez'

SELECT * FROM emp WHERE lname = 'Olagappan'

SELECT * FROM emp WHERE lname = 'Wu'

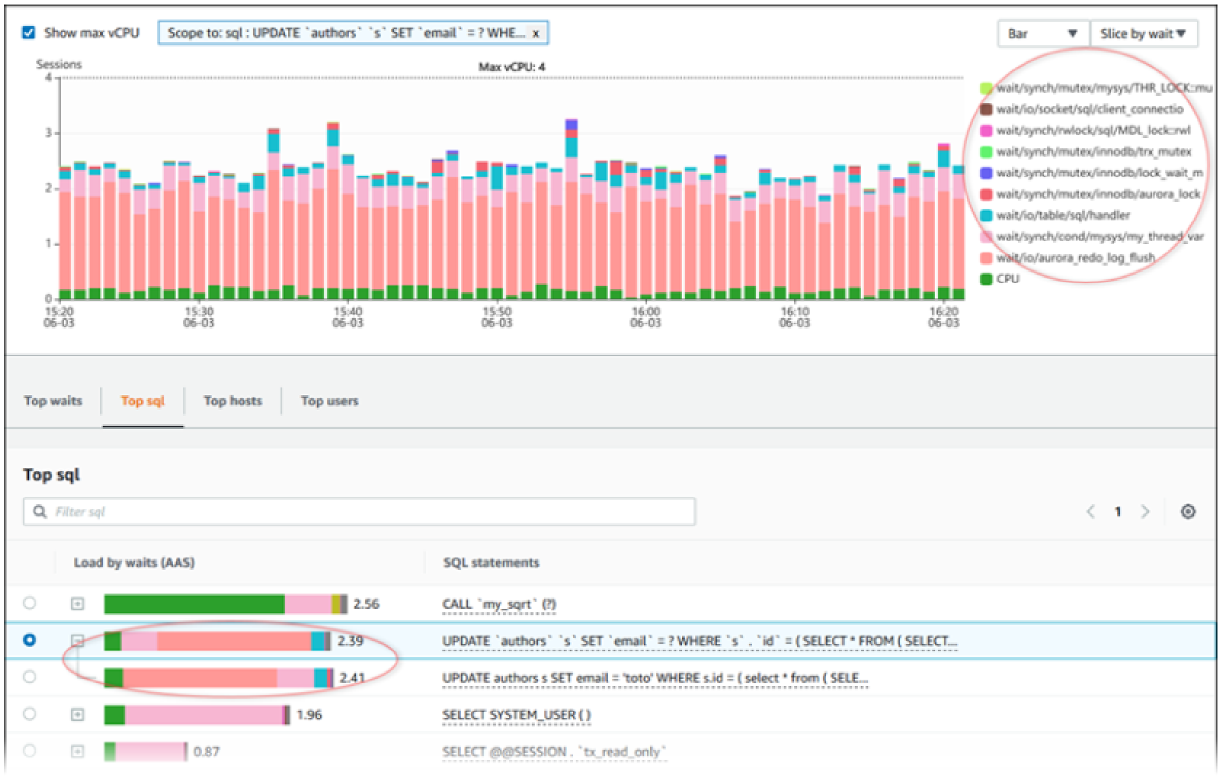
To see the literal SQL statements in a digest, select the query, and then choose the plus symbol (+). In the following example, the selected query is a digest.



Load by waits (AAS)	SQL statements
0.88	select minute_rollups(?)
0.50	select minute_rollups(1000000)
0.53	select count(*) from authors where ic

In Top SQL, the Load by waits (AAS) column illustrates the percentage of the database load associated with each top load item. This column reflects the load for that item by whatever grouping is currently selected in the DB Load Chart. For more information about Average active sessions (AAS)

For example, you might group the DB load chart by wait states. You examine SQL queries in the top load items table. In this case, the DB Load by Waits bar is sized, segmented, and color-coded to show how much of a given wait state that query is contributing to. It also shows which wait states are affecting the selected query.

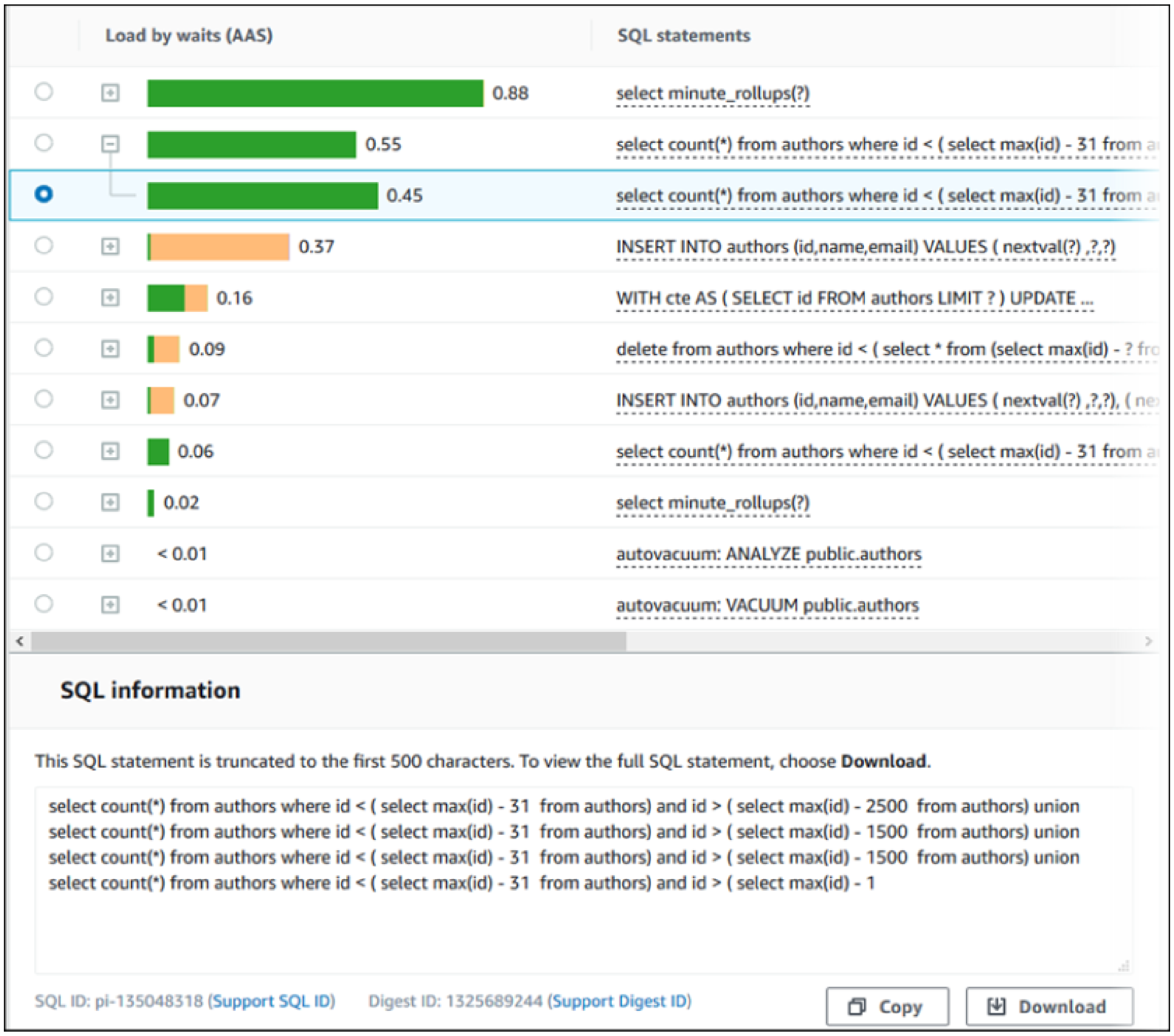


🟩	wait/synch/mutex/mysys/THR_LOCK::mu
🟫	wait/io/socket/sql/client_connectio
🟪	wait/synch/rwlock/sql/MDL_lock::rwl
🟩	wait/synch/mutex/innodb/trx_mutex
🟦	wait/synch/mutex/innodb/lock_wait_m
🟥	wait/synch/mutex/innodb/aurora_lock
🟦	wait/io/table/sql/handler
🟪	wait/synch/cond/mysys/my_thread_var
🟥	wait/io/aurora_redo_log_flush
🟩	CPU

Top sql

Load by waits (AAS)	SQL statements
2.56	CALL `my_sqrt` (?)
2.39	UPDATE `authors` `s` SET `email` = ? WHERE `s` . `id` = (SELECT * FROM (SELECT...
2.41	UPDATE authors s SET email = 'toto' WHERE s.id = (select * from (SELE...
1.96	SELECT SYSTEM_USER ( )
0.87	SELECT @@SESSION . `tx_read_only`

In the Top SQL table, you can open a statement to view its information. The information appears in the bottom pane.



Load by waits (AAS)	SQL statements
0.88	select minute_rollups(?)
0.55	select count(*) from authors where id < (select max(id) - 31 from a
0.45	select count(*) from authors where id < (select max(id) - 31 from a
0.37	INSERT INTO authors (id,name,email) VALUES (nextval(?) ,?,?)
0.16	WITH cte AS (SELECT id FROM authors LIMIT ?) UPDATE ...
0.09	delete from authors where id < (select * from (select max(id) - ? fro
0.07	INSERT INTO authors (id,name,email) VALUES (nextval(?) ,?,?), (ne
0.06	select count(*) from authors where id < (select max(id) - 31 from a
0.02	select minute_rollups(?)
< 0.01	autovacuum: ANALYZE public.authors
< 0.01	autovacuum: VACUUM public.authors

SQL information

This SQL statement is truncated to the first 500 characters. To view the full SQL statement, choose **Download**.

```
select count(*) from authors where id < ( select max(id) - 31  from authors) and id > ( select max(id) - 2500  from authors) union
select count(*) from authors where id < ( select max(id) - 31  from authors) and id > ( select max(id) - 1500  from authors) union
select count(*) from authors where id < ( select max(id) - 31  from authors) and id > ( select max(id) - 1500  from authors) union
select count(*) from authors where id < ( select max(id) - 31  from authors) and id > ( select max(id) - 1
```

SQL ID: pi-135048318 (Support SQL ID)   Digest ID: 1325689244 (Support Digest ID)