Simple overview of what DPQL can do, with examples for each:

Kim - 2026-04-28 - Reacties (0) - Using Deskpro

Aggregation (KPIs & totals)

Used for counts and summary stats.

```
SELECT DPQL_COUNT() AS 'Total Tickets'
FROM tickets
WHERE tickets.status = 'resolved'
```

Grouping & reporting

Used to break data down (e.g. by agent, status, etc.)

```
SELECT DPQL_COUNT() AS 'Total Tickets'
FROM tickets
GROUP BY tickets.agent AS 'Agent'
```

You can use an alias in the Group By section.

Matrix (pivot-style reports)

Used for cross-tab reports (e.g. agent vs status)

```
SELECT DPQL_COUNT()
FROM tickets
GROUP BY DPQL_MATRIX(tickets.agent, tickets.status)
```

Filtering data

Used to limit results

```
SELECT DPQL_COUNT() AS 'Open Tickets'
FROM tickets
WHERE tickets.status IN ('pending', 'awaiting_agent', 'awaiting_user')
```

Conditional values (IF())

Used to change how values are displayed

```
SELECT  DPQL_COUNT() AS 'stat_value', IF(DPQL_COUNT() = 0, 'No tickets found',
'Tickets created') AS 'stat_description'
FROM tickets
WHERE tickets.date_created = ${date}
```

Basic calculations

Used for things like time differences

```
SELECT
  tickets.id,
  (UNIX_TIMESTAMP(tickets.date_resolved) - UNIX_TIMESTAMP(tickets.date_created)) /
3600 AS 'Hours to Resolve'
FROM tickets
WHERE tickets.status = 'resolved'
```

DPQL_PERCENT() calculate percentages

Example: % of tickets that are resolved

```
SELECT DPQL_PERCENT(tickets.status = 'resolved') AS 'Resolved %'
FROM tickets
WHERE tickets.date_created = %THIS_MONTH%
```

What this does:

Counts how many rows match the condition

Divides by total rows

Returns a percentage

SUM() adds numbers together

```
SELECT SUM(tickets.urgency) AS 'Total Urgency'
FROM tickets
WHERE tickets.status = 'resolved'
```

This returns:

urgency 3 + urgency 2 + urgency 5 = 10

So instead of counting tickets, it adds up a numeric field across them.

Working with dates

Used for grouping or filtering by time

```
SELECT DPQL_COUNT()
FROM tickets
WHERE tickets.date_created = %THIS_MONTH%
```

You can find the full list of date placeholders in this section of the reports guide: List of Date Placeholders

Date formatting functions

Example: group tickets by month

```
SELECT DPQL_COUNT()
FROM tickets
GROUP BY DPQL_ALIAS(DATE_FORMAT(tickets.date_created, '%Y-%m'), 'Month')
```

🔗 Cross-referencing related data

Access related fields (no JOIN needed)

```
SELECT
  tickets.id,
  tickets.ticket_slas.sla_status AS 'SLA Status'
FROM tickets
```

You can read more about cross-referencing in this section of the reports guide: Cross-referencing Table Fields

🔗 DPQL_CONCAT() — combine text

Example: combine ticket ID + subject

```
SELECT
  DPQL_CONCAT('#', tickets.id, ' - ', tickets.subject) AS 'Ticket'
FROM tickets
```

Real use:

Build readable labels

Combine fields for exports

Format output nicely

🔗 Simple subquery

Example: tickets that have feedback

```
SELECT tickets.id
FROM tickets
WHERE tickets.id IN (
  SELECT ticket_feedback.ticket_id
  FROM ticket_feedback
)
```

Subquery is in WHERE

It's simple (no nesting, no aggregates inside aggregates)

Other useful DPQL functions

🔗 DPQL_COUNT_DISTINCT() count unique values

Counts how many different values exist in a field.

```
SELECT DPQL_COUNT_DISTINCT(tickets.agent) AS 'Unique Agents'
FROM tickets
```

Useful for: "how many agents / orgs / categories are involved?"

🔗 DPQL_NOW() current date & time

Returns full timestamp.

```
SELECT
  tickets.id,
  (UNIX_TIMESTAMP(DPQL_NOW()) - UNIX_TIMESTAMP(tickets.date_created)) / 3600 AS 'Hours
Open'
FROM tickets
```

Now it answers: “How long has each ticket been open right now?”

Useful for: time comparisons

DPQL_DAYNAME() weekday name

Returns the day (e.g. Monday, Tuesday).

```
SELECT DPQL_COUNT()
FROM tickets
GROUP BY DPQL_DAYNAME(tickets.date_created) AS 'Day'
```

Useful for: “which day is busiest?”

DPQL_DAYOFMONTH() day number (1–31)

```
SELECT DPQL_COUNT()
FROM tickets
GROUP BY DPQL_DAYOFMONTH(tickets.date_created) AS 'Day'
```

Useful for: daily trends within a month

DPQL_DAYOFWEEK() weekday number (1–7)

```
SELECT DPQL_COUNT()
FROM tickets
GROUP BY DPQL_DAYOFWEEK(tickets.date_created) AS 'Day #'
```

Useful for: sorting days numerically

DPQL_HOUR() hour of day (0–23)

```
SELECT DPQL_COUNT()
FROM tickets
GROUP BY DPQL_HOUR(tickets.date_created) AS 'Hour'
```

Useful for: “what time of day are tickets created?”

DPQL_MINUTE() minute (0–59)

```
SELECT DPQL_COUNT()
FROM tickets
GROUP BY DPQL_MINUTE(tickets.date_created) AS 'Minute'
```

Rarely used, but helpful for very granular analysis

DPQL_MONTH() month number (1–12)

```
SELECT DPQL_COUNT()
FROM tickets
```

```
GROUP BY DPQL_MONTH(tickets.date_created) AS 'Month'
```

Useful for: yearly breakdowns

DPQL_MONTHNAME() month name

```
SELECT DPQL_COUNT()
FROM tickets
GROUP BY DPQL_MONTHNAME(tickets.date_created) AS 'Month'
```

Same as above, but readable labels

DPQL_YEAR() extract year

```
SELECT DPQL_COUNT()
FROM tickets
GROUP BY DPQL_YEAR(tickets.date_created) AS 'Year'
```

Useful for: multi-year reporting

DPQL_DATE_OFFSET_GROUP() group time differences

Groups time differences into ranges (e.g. 0–15 min, 1–2 hours).

```
SELECT DPQL_COUNT()
FROM tickets
WHERE tickets.status = 'resolved'
GROUP BY DPQL_DATE_OFFSET_GROUP(tickets.date_resolved, tickets.date_created) AS 'Time
to Resolve'
```

Very useful for: SLA / resolution time reporting

DPQL_TIME_LENGTH() format duration

Turns seconds into readable time (e.g. “2h 30m”).

```
SELECT
  tickets.id,
  DPQL_TIME_LENGTH(UNIX_TIMESTAMP(tickets.date_resolved) -
UNIX_TIMESTAMP(tickets.date_created)) AS 'Resolution Time'
FROM tickets
```

Useful for: displaying durations cleanly

DPQL_FORMAT() format values

```
SELECT
  tickets.id AS 'Ticket ID',
  DPQL_FORMAT(tickets.date_created, 'date') AS 'Created Date'
FROM tickets
```

It takes a full date/time value like:

2026-04-28 14:35:22

and displays it as a cleaner date value.

🌍 DPQL_TO_UTC() convert to UTC

Converts a date to UTC timezone.

```
SELECT DPQL_TO_UTC(tickets.date_created) AS 'Created UTC'
FROM tickets
```

🌍 DPQL_UTC() run calculation in UTC

Forces date logic to use UTC.

```
SELECT DPQL_COUNT()
FROM tickets
WHERE DPQL_UTC(tickets.date_created) = %THIS_MONTH%
```

Useful for: consistency across time zones

What DPQL does NOT support

- No JOIN
- No HAVING
- No complex subqueries
- No scripting logic (no Twig-style if/else)
- No dynamic query structure

You can find the list of supported functions in the reports guide here: List of Functions

In short: DPQL is designed for reporting (counts, grouping, filtering), with some light formatting using functions like IF().

If you have a specific report in mind, feel free to reach out to support and we can help build it.