

SQL Subqueries Guide PDF Page

SQL Subqueries Guide PDF Page: Content + <head> (Uncodemy)

FINAL URL: <https://uncodemy.com/sql-subqueries-guide-pdf>

Publish this page at exactly this URL (non-www, no trailing slash). Canonical, og:url and sitemap entry must match it.

Prepared: 26 October 2026 · **Primary keyword:** `sql subqueries guide pdf` · **Search volume:** not checked (keyword tool out of credits); verify in Search Console.

Read first (5 flags):

- Your course PDF title, page count and size were not shared.** Download cards are fill-in slots (`[CONFIRM]`). Card 1 is your course PDF; cards 2 and 3 only if you have more files.
- One unique file per button.** The same course PDF repeated on many pages is fine as a resource, but each page's text must stay unique and useful without the file.
- Free or gated?** Not stated. The page says "Download", not "Free". If a form is used, add a consent checkbox and Privacy Policy link (DPDP Act 2023).
- No made-up stats or testimonials;** no placement or salary claims without a named, dated source.
- Author and reviewer are placeholders;** use real named trainers and one real "Last reviewed" date.

PART 1: Page plan

Item	Decision
Final URL	<code>https://uncodemy.com/sql-subqueries-guide-pdf</code>
Intent (anti-cannibalisation)	SUBQUERY TYPES AND PITFALLS. CTEs have their own page; joins are covered in the SQL for analytics guide.
Why not thin	Explainer-style sections, FAQs and cross-links make the page useful even without a download
Schema	WebSite, Organization, Breadcrumb, CollectionPage, FAQPage (only FAQs visible on the page). PDF <code>ItemList</code> provided commented, to enable after titles are final
Share image	<code>https://uncodemy.com/image/logo-og.jpg</code> for now; custom 1200x630 card recommended

PART 2: Page content (copy-paste ready)

SEO Title (35 chars):

SQL Subqueries Guide PDF | Uncodemy

Meta Description (136 chars):

Download the SQL subqueries guide PDF from Uncodemy: scalar, IN, NOT IN, correlated and derived-table subqueries, plus join comparisons.

H1: SQL Subqueries Guide PDF Downloads **Breadcrumb:** Home > SQL Subqueries Guide PDF **Final URL:** `https://uncodemy.com/sql-subqueries-guide-pdf`

Byline: Prepared by the Uncodemy trainer team `[CONFIRM: author name and role]` · Reviewed by `[CONFIRM: reviewer name and role]` · Last reviewed: `[DATE]`

Quick Answer

This page offers Uncodemy's SQL subqueries guide as a PDF download. A subquery is a query nested inside another. The guide explains scalar subqueries, IN and EXISTS, correlated subqueries and derived tables, shows when a join or a CTE is a better choice and points out the NOT IN trap with nulls.

Download SQL Subqueries Guide PDF

`[CONFIRM: free or gated, and whether details are needed before download]`

PDF 1 (main): `[CONFIRM TITLE: your Data Analytics course PDF]`
`[CONFIRM: one line on what it covers and who it is for]` · `[N]` pages · `[X]` MB · PDF · Updated `[DATE]`
[Download PDF] → `https://uncodemy.com/pdf/sql-subqueries-guide.pdf` (*your course PDF, uploaded under this file name*)

PDF 2 (optional): `[CONFIRM TITLE]`
Only if you have a second PDF that genuinely fits this topic. **[Download PDF]** → `https://uncodemy.com/pdf/sql-subqueries-guide-2.pdf`

PDF 3 (optional): [CONFIRM TITLE]

Same rule. [Download PDF] → <https://uncodemy.com/pdf/sql-subqueries-guide-3.pdf>

Each button must open its own file. Do not point several buttons at one file. Delete unused optional cards.

What is a subquery?

A subquery is a SELECT statement placed inside another statement, usually in brackets. The inner query runs to produce a value, a list of values or a table, and the outer query uses that result. Subqueries let you answer questions in two steps within a single statement, for example: find the orders whose total is above the average order total. The inner query calculates the average, and the outer query compares each order with it. The examples use these tables: customers(customer_id, name, city, signup_date), orders(order_id, customer_id, order_date, status, order_total), order_items(order_id, product_id, quantity, unit_price) and products(product_id, product_name, category). Dates are text in YYYY-MM-DD form, so the examples run in SQLite and need small changes in other databases.

Types of subqueries

Type	Returns	Typical use
Scalar subquery	One value	Compare each row with an overall average
Multi-row subquery with IN	A list of values	Filter by membership
EXISTS subquery	True if any row is found	Check whether related rows exist
Correlated subquery	Depends on the outer row	Compare each row with its own group
Derived table (in FROM)	A table	Aggregate first, then filter or join

A scalar subquery

Orders whose total is above the average of all non-cancelled orders.

```
SELECT order_id, customer_id, order_total
FROM orders
WHERE status <> 'cancelled'
      AND order_total > (SELECT AVG(order_total)
                        FROM orders
                        WHERE status <> 'cancelled')
ORDER BY order_total DESC;
```

IN and NOT IN

Customers who have placed at least one order, and customers who never have. Note the safer NOT EXISTS form below.

```
-- customers with at least one order
SELECT customer_id, name
FROM customers
WHERE customer_id IN (SELECT customer_id FROM orders);

-- customers with no orders (NOT EXISTS is null-safe)
SELECT c.customer_id, c.name
FROM customers c
WHERE NOT EXISTS (SELECT 1 FROM orders o WHERE o.customer_id = c.customer_id);
```

The NOT IN trap with NULL values

NOT IN behaves surprisingly if the subquery returns even one NULL. Because a comparison with NULL is unknown, not true, the whole NOT IN condition becomes unknown for every row and the query can return nothing at all, with no error. If a column can be NULL, prefer NOT EXISTS, or add a filter such as WHERE column IS NOT NULL inside the subquery. This is a classic interview question and a real source of silent errors in reports.

A correlated subquery

Each order is compared with the average of that same customer's orders. The inner query refers to the outer row, so it runs logically once per outer row.

```
SELECT o.order_id, o.customer_id, o.order_total
FROM orders o
WHERE o.status <> 'cancelled'
      AND o.order_total > (SELECT AVG(o2.order_total)
                        FROM orders o2
                        WHERE o2.customer_id = o.customer_id
                        AND o2.status <> 'cancelled');
```

A derived table in FROM

Aggregate first in a subquery, then join the result to another table.

```
SELECT c.city, t.orders, t.revenue
FROM (
  SELECT customer_id, COUNT(*) AS orders, SUM(order_total) AS revenue
  FROM orders
  WHERE status <> 'cancelled'
  GROUP BY customer_id
) t
JOIN customers c ON c.customer_id = t.customer_id
ORDER BY t.revenue DESC;
```

Subquery or join or CTE?

- Use a join when you need columns from both tables in the result.
- Use EXISTS or IN when you only need to test for membership or existence.
- Use a scalar subquery for a single overall value such as an average.
- Use a CTE when the logic has several steps or is reused.
- Use a window function when you need group values beside each row without collapsing rows.

Common subquery mistakes

- **A scalar subquery that returns more than one row**, which causes an error.
- **NOT IN with NULLs**, which returns no rows.
- **Correlated subqueries on huge tables**, which can be slow compared with a join.
- **Forgetting aliases** for derived tables, which many databases require.
- **Unclear nesting** that a CTE would make readable.

Practice tasks

Write queries for these questions: customers whose total spend is above the average customer spend; products never ordered; the most recent order of each customer using a correlated subquery; and the number of customers per city using a derived table. Then rewrite each with a join or a CTE and compare which is clearer. Understanding several ways to express the same idea is what makes SQL feel natural.

Subquery versus join: which to use

Situation	Prefer	Reason
Need columns from both tables in the result	Join	A subquery in WHERE cannot return its columns to the outer query
Only filtering by existence	EXISTS subquery	Stops at the first match and reads clearly
Compare with one aggregate value	Scalar subquery	Short and readable
Complex multi-step logic	CTE	Named steps are easier to read and debug
Find rows with no match	NOT EXISTS or LEFT JOIN with IS NULL	Avoids the NOT IN problem with NULL values

Performance notes

- Modern optimisers often rewrite subqueries and joins into the same plan, so write the clearer version first.
- Correlated subqueries run logically once per outer row; check the execution plan if the query is slow.
- Index the columns used in the correlation and join conditions.
- Measure on realistic data volumes instead of guessing.

Related Uncodemy resources

- [SQL CTE Guide PDF](#)
- [SQL for Data Analytics Guide PDF](#)
- [SQL Analytics Functions PDF](#)
- [SQL for Data Analysts PDF](#)

Learn with a trainer [use only the line that is true]

- **Data Analytics course:** <https://uncodemy.com/course/data-analytics-training-course-in-noida>. URL verified on the live site.

(One soft call to action: course link or free demo. No pop-up.)

About these PDFs

These PDFs are study material for learning and revision. They are not a certificate, a degree or a promise of a job, and technology changes, so check official documentation for the latest details. Spot an error? Write to [CONFIRM: feedback_email] and the trainer team will review it.

Frequently Asked Questions

What is a subquery in SQL?

A subquery is a query nested inside another query. The inner query produces a value, list or table that the outer query uses.

What is the difference between IN and EXISTS?

IN compares a value with a list returned by a subquery. EXISTS checks whether the subquery returns any row. EXISTS is also safer with NULLs when negated.

What is a correlated subquery?

A subquery that refers to columns from the outer query, so it is evaluated in relation to each outer row.

Why does NOT IN sometimes return no rows?

If the subquery returns a NULL, comparisons become unknown and NOT IN can return nothing. Use NOT EXISTS or filter out NULLs.

When should I use a join instead of a subquery?

Use a join when you need columns from both tables, and a subquery when you only need to filter or test existence.

Are these PDFs free to download?

[CONFIRM: state clearly whether the PDFs are free, or whether a short form is needed first.]

The last FAQ contains [CONFIRM], so it is NOT in the FAQ schema. Fill it and keep it on the page, or delete it.

PART 3: <head> section (copy-paste ready)

Final URL: <https://uncodemy.com/sql-subqueries-guide-pdf>

```
<!-- =====
UNCODEMY <head> | FINAL URL: https://uncodemy.com/sql-subqueries-guide-pdf
Prepared 26 October 2026. Paste into the page <head>; keep your site-wide favicon, fonts, analytics.
BEFORE GO-LIVE: (1) set dateModified in the JSON-LD to the real publish date;
(2) replace og:image with a 1200x630 share card when available;
(3) once PDF titles are final, uncomment the ItemList block at the bottom.
===== -->
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1">

<title>SQL Subqueries Guide PDF | Uncodemy</title>
<meta name="description" content="Download the SQL subqueries guide PDF from Uncodemy: scalar, IN, NOT IN, correlated and derived-table subqueries, plus join
comparisons.">
<link rel="canonical" href="https://uncodemy.com/sql-subqueries-guide-pdf">
<meta name="robots" content="index, follow, max-snippet:-1, max-image-preview:large, max-video-preview:-1">
<meta name="author" content="Uncodemy">
<meta name="publisher" content="Uncodemy Edutech Pvt. Ltd.">
<meta name="language" content="en-IN">
<link rel="alternate" hreflang="en-in" href="https://uncodemy.com/sql-subqueries-guide-pdf">
<link rel="alternate" hreflang="x-default" href="https://uncodemy.com/sql-subqueries-guide-pdf">

<!-- Open Graph -->
<meta property="og:type" content="website">
<meta property="og:site_name" content="Uncodemy">
<meta property="og:locale" content="en_IN">
<meta property="og:url" content="https://uncodemy.com/sql-subqueries-guide-pdf">
<meta property="og:title" content="SQL Subqueries Guide PDF | Uncodemy">
<meta property="og:description" content="Download the SQL subqueries guide PDF from Uncodemy: scalar, IN, NOT IN, correlated and derived-table subqueries, pl
join comparisons.">
<meta property="og:image" content="https://uncodemy.com/image/logo-og.jpg">
<meta property="og:image:alt" content="Uncodemy SQL Subqueries Guide PDF study material">

<!-- X (Twitter) -->
<meta name="twitter:card" content="summary_large_image">
<meta name="twitter:site" content="@uncodemy">
<meta name="twitter:title" content="SQL Subqueries Guide PDF | Uncodemy">
<meta name="twitter:description" content="Download the SQL subqueries guide PDF from Uncodemy: scalar, IN, NOT IN, correlated and derived-table subqueries, p
join comparisons.">
<meta name="twitter:image" content="https://uncodemy.com/image/logo-og.jpg">
<meta name="twitter:image:alt" content="Uncodemy SQL Subqueries Guide PDF study material">

<!-- Structured data: one @graph (WebSite, Organization, Breadcrumb, CollectionPage, FAQPage) -->
<script type="application/ld+json">
{
  "@context": "https://schema.org",
  "@graph": [
    {
      "@type": "WebSite",
      "@id": "https://uncodemy.com/#website",
      "url": "https://uncodemy.com/",
      "name": "Uncodemy",
      "publisher": {
        "@id": "https://uncodemy.com/#organization"
      },
      "inLanguage": "en-IN"
    },
    {
      "@type": "EducationalOrganization",
      "@id": "https://uncodemy.com/#organization",
      "name": "Uncodemy",
      "legalName": "Uncodemy Edutech Pvt. Ltd.",
      "url": "https://uncodemy.com/",
      "logo": {
        "@type": "ImageObject",
        "url": "https://uncodemy.com/uc-layout/img/Logo.webp"
      },
      "email": "info@uncodemy.com",
      "identifier": {
        "@type": "PropertyValue",
        "name": "CIN",
        "value": "U85499UP2024PTC195902"
      },
      "contactPoint": [
        {
          "@type": "ContactPoint",
          "telephone": "+91-9818366550",
          "contactType": "admissions",
          "areaServed": "IN",
          "availableLanguage": [
            "en",
            "hi"
          ]
        }
      ],
      "sameAs": [
        "https://x.com/uncodemy",
        "https://www.facebook.com/uncodemyofficial/",
        "https://www.linkedin.com/company/uncodemy/",
        "https://www.instagram.com/uncodemyofficial/"
      ]
    },
    {
      "@type": "BreadcrumbList",
      "@id": "https://uncodemy.com/sql-subqueries-guide-pdf#breadcrumb",
      "itemListElement": [
        {
          "@type": "ListItem",
          "position": 1,
          "name": "Home",
          "item": "https://uncodemy.com/"
        }
      ]
    }
  ]
}
```

```

    },
    {
      "@type": "ListItem",
      "position": 2,
      "name": "SQL Subqueries Guide PDF",
      "item": "https://uncodemy.com/sql-subqueries-guide-pdf"
    }
  ]
},
{
  "@type": "CollectionPage",
  "@id": "https://uncodemy.com/sql-subqueries-guide-pdf#webpage",
  "url": "https://uncodemy.com/sql-subqueries-guide-pdf",
  "name": "SQL Subqueries Guide PDF Downloads",
  "description": "Download the SQL subqueries guide PDF from Uncodemy: scalar, IN, NOT IN, correlated and derived-table subqueries, plus join comparisons",
  "inLanguage": "en-IN",
  "isPartOf": {
    "@id": "https://uncodemy.com/#website"
  },
  "publisher": {
    "@id": "https://uncodemy.com/#organization"
  },
  "breadcrumb": {
    "@id": "https://uncodemy.com/sql-subqueries-guide-pdf#breadcrumb"
  },
  "about": {
    "@type": "Thing",
    "name": "SQL Subqueries Guide"
  },
  "primaryImageOfPage": {
    "@type": "ImageObject",
    "url": "https://uncodemy.com/image/logo-og.jpg"
  },
  "dateModified": "2026-10-26"
},
{
  "@type": "FAQPage",
  "@id": "https://uncodemy.com/sql-subqueries-guide-pdf#faq",
  "inLanguage": "en-IN",
  "isPartOf": {
    "@id": "https://uncodemy.com/sql-subqueries-guide-pdf#webpage"
  },
  "mainEntity": [
    {
      "@type": "Question",
      "name": "What is a subquery in SQL?",
      "acceptedAnswer": {
        "@type": "Answer",
        "text": "A subquery is a query nested inside another query. The inner query produces a value, list or table that the outer query uses."
      }
    },
    {
      "@type": "Question",
      "name": "What is the difference between IN and EXISTS?",
      "acceptedAnswer": {
        "@type": "Answer",
        "text": "IN compares a value with a list returned by a subquery. EXISTS checks whether the subquery returns any row. EXISTS is also safer with NU
when negated."
      }
    },
    {
      "@type": "Question",
      "name": "What is a correlated subquery?",
      "acceptedAnswer": {
        "@type": "Answer",
        "text": "A subquery that refers to columns from the outer query, so it is evaluated in relation to each outer row."
      }
    },
    {
      "@type": "Question",
      "name": "Why does NOT IN sometimes return no rows?",
      "acceptedAnswer": {
        "@type": "Answer",
        "text": "If the subquery returns a NULL, comparisons become unknown and NOT IN can return nothing. Use NOT EXISTS or filter out NULLs."
      }
    },
    {
      "@type": "Question",
      "name": "When should I use a join instead of a subquery?",
      "acceptedAnswer": {
        "@type": "Answer",
        "text": "Use a join when you need columns from both tables, and a subquery when you only need to filter or test existence."
      }
    }
  ]
}
]
}
}
</script>

<!-- ADD AFTER THE PDF TITLES ARE FINAL (remove this wrapper, fill every [...] value):
<script type="application/ld+json">
{
  "@context": "https://schema.org",
  "@type": "ItemList",
  "@id": "https://uncodemy.com/sql-subqueries-guide-pdf#pdfs",
  "name": "SQL Subqueries Guide PDF downloads",
  "numberOfItems": 3,
  "itemListElement": [
    {
      "@type": "ListItem",
      "position": 1,
      "item": {
        "@type": "DigitalDocument",
        "name": "[PDF 1 TITLE]",
        "description": "[PDF 1 ONE-LINE DESCRIPTION]",
        "encodingFormat": "application/pdf",
        "url": "https://uncodemy.com/pdf/sql-subqueries-guide.pdf",
        "inLanguage": "en",
        "publisher": {
          "@id": "https://uncodemy.com/#organization"
        }
      }
    }
  ]
}
}
}

```

```

    }
  },
  {
    "@type": "ListItem",
    "position": 2,
    "item": {
      "@type": "DigitalDocument",
      "name": "[PDF 2 TITLE]",
      "description": "[PDF 2 ONE-LINE DESCRIPTION]",
      "encodingFormat": "application/pdf",
      "url": "https://uncodemy.com/pdf/sql-subqueries-guide-2.pdf",
      "inLanguage": "en",
      "publisher": {
        "@id": "https://uncodemy.com/#organization"
      }
    }
  },
  {
    "@type": "ListItem",
    "position": 3,
    "item": {
      "@type": "DigitalDocument",
      "name": "[PDF 3 TITLE]",
      "description": "[PDF 3 ONE-LINE DESCRIPTION]",
      "encodingFormat": "application/pdf",
      "url": "https://uncodemy.com/pdf/sql-subqueries-guide-3.pdf",
      "inLanguage": "en",
      "publisher": {
        "@id": "https://uncodemy.com/#organization"
      }
    }
  }
]
}
</script>
-->

```

DEVELOPER CHECKLIST

- ☐ Final URL live at `https://uncodemy.com/sql-subqueries-guide-pdf`; canonical, og:url and sitemap identical.
- ☐ Each button: `<a href="/pdf/sql-subqueries-guide.pdf" download type="application/pdf">`, its own file, size shown.
- ☐ If a form precedes download: consent checkbox, Privacy Policy link, nothing pre-ticked.
- ☐ One `<h1>`; real HTML tables; code samples in `<pre><code>`.
- ☐ Remove any `<meta name="keywords">` from the template.
- ☐ Test in Google Rich Results Test and validator.schema.org after publishing.
- ☐ Add to sitemap and any PDF/resources hub.

STILL NEEDED FROM YOU

#	Item
1	Course PDF title, description, page count, size, and the file URL for each page
2	Free or gated (and form fields if gated)
3	Author, reviewer, publish and last-reviewed dates
4	Course URL for the CTA (lines marked <code>[CONFIRM]</code>)
5	Feedback email
6	Confirmation that Uncodemy owns the rights to every PDF