



# Vollständiger

# SQL Bootcamp

# Udemy Kurs-Notizen Vollständiger SQL Bootcamp





## Inhaltsverzeichnis:

- SELECT
- SELECT DISTINCT
- WHERE
- PostgreSQL WHERE Beispiele
- LIMIT
- IN Operator
- PostgreSQL IN Operator Beispiele
- NOT IN Operator
- ORDER BY
- PostgreSQL ORDER BY Operator Beispiele
- BETWEEN
- PostgreSQL BETWEEN Beispiele
- LIKE
- GROUP BY
- PostgreSQL GROUP BY mit der SUM-Funktion Beispiele
- HAVING
- Beispiel
- JOINS
- SUBQUERY (UNTERABFRAGE)
- CREATE TABLE und Einschränkungen
- PostgreSQL Spalteneinschränkungen
- PostgreSQL Tabelleneinschränkungen
- PostgreSQL CREATE TABLE Beispiel



# Select

010010000111  
0010100110100  
1000101100110  
010010000111  
1011001100110  
0010100110100

Eine der häufigsten Aufgaben bei der Arbeit mit PostgreSQL besteht darin, Daten aus Tabellen mithilfe der SELECT-Anweisung abzufragen. Die SELECT-Anweisung ist eine der komplexesten Anweisungen in PostgreSQL. Es gibt viele Klauseln, die man kombinieren kann, um eine leistungsstarke Abfrage zu bilden.

Aufgrund seiner Komplexität teilen wir das PostgreSQL-SELECT-Anweisung-Tutorial in viele kurze Tutorials auf, so dass du jede Klausel der SELECT-Anweisung einfacher lernen kannst.

Die folgenden Klauseln erscheinen in der SELECT-Anweisung:

- Unter Verwendung des DISTINCT-Operators verschiedene Zeilen auswählen
- Zeilen mit der WHERE-Klausel filtern
- Zeilen mit der ORDER-BY Klausel sortieren
- Zeilen basierend auf verschiedenen Operatoren wie BETWEEN, IN und LIKE auswählen
- Gruppieren von Zeilen in Gruppen mithilfe der GROUP BY-Klausel
- Anwenden von Bedingungen für Gruppen mithilfe der HAVING-Klausel
- Eine andere Tabelle mithilfe der INNER JOIN, LEFT JOIN, RIGHT JOIN -Klauseln verknüpfen

Beginnen wir mit einer grundlegenden Form der **SELECT**-Anweisung, um Daten aus einer Tabelle abzufragen.

Im Folgenden wird die Syntax der **SELECT**-Anweisung veranschaulicht:

```
1 SELECT column_1, column_2,  
2 FROM table name
```

Lass uns die **SELECT**-Anweisung detailliert untersuchen:

- Zuerst gibt man eine Liste von Spalten in der Tabelle an, aus der man Daten in der **SELECT**-Klausel abfragen möchte. Man verwendet ein Komma zwischen jeder Spalte, wenn man Daten aus mehreren Spalten abfragen möchte. Wenn man Daten aus allen Spalten abfragen will, kann man ein Sternchen (\*) als Kurzschrift für alle Spalten verwenden.

- Zweitens gibt man den Tabellennamen nach dem Schlüsselwort **FROM** an

Beachte, dass die SQL-Sprache nicht zwischen Gross- und Kleinschreibung unterscheidet. Das bedeutet, egal ob man **SELECT** oder **select** verwendet, der Effekt ist der gleiche. Wir werden SQL-Schlüsselwörter in Großbuchstaben verwenden, um den Code leichter lesbar und klar zu machen.

## SELECT DISTINCT

- Die DISTINCT-Klausel wird in der SELECT-Anweisung verwendet, um doppelte Zeilen aus einer Ergebnismenge zu entfernen. Die DISTINCT-Klausel behält eine Zeile für jede Gruppe von Duplikaten bei. Man kann die DISTINCT-Klausel für eine oder mehrere Spalten einer Tabelle verwenden.

Die Syntax der DISTINCT-Klausel lautet wie folgt:

```
1 SELECT DISTINCT column_1  
2 FROM table_name;
```

- Wenn man mehrere Spalten angibt, wertet die DISTINCT-Klausel das Duplikat basierend auf der Kombination der Werte dieser Spalten aus.

```
1 SELECT DISTINCT column_1, column_2  
2 FROM table_name;
```

- PostgreSQL stellt auch DISTINCT ON (Ausdruck) zur Verfügung, um die “erste” Zeile jeder Gruppe von Duplikaten zu behalten, wo der Ausdruck gleich ist. Siehe folgende Syntax:

```
1 SELECT DISTINCT ON (column_1), column_2  
2 FROM table_name;  
3 ORDER BY column_1,column2;
```

- Die Reihenfolge der von der SELECT-Anweisung zurückgegebenen Zeilen ist nicht vorhersagbar, daher ist auch die “erste” Zeile jeder Gruppe des Duplikats nicht vorhersagbar. Es ist eine gute Übung, immer die ORDER BY-Klausel mit dem DISTINCT ON (Ausdruck) zu verwenden, um das Ergebnis offensichtlich zu machen.
- Beachte, dass der Ausdruck DISTINCT ON mit dem Ausdruck ganz links in der Klausel ORDER BY übereinstimmen muss.

## WHERE

- Die Syntax der PostgreSQL WHERE-Klausel ist wie folgt:

1 `SELECT column_1, column_2 ... column_n`

2 `FROM table name`

3 `WHERE conditions;`

- Die WHERE-Klausel wird unmittelbar nach der FROM-Klausel der SELECT-Anweisung angezeigt. Die Bedingungen werden verwendet, um die Zeilen zu filtern, die von der SELECT-Anweisung zurückgegeben werden. PostgreSQL bietet verschiedene Standardoperatoren zum Erstellen der Bedingungen.

Die folgende Tabelle zeigt die Standardvergleichsoperatoren.

=	gleich
>	größer als
<	kleiner als
>=	größer als oder gleich

<code>&lt;=</code>	kleiner als oder gleich
<code>&lt;&gt; or !=</code>	ungleich
<code>AND</code>	Logischer Operator UND
<code>OR</code>	größer als oder gleich

Lass uns mit einigen Beispielen zur Verwendung der WHERE-Klausel mit Bedingungen üben.

### PostgreSQL WHERE-Beispiele

Wenn man alle Kunden, dessen Vornamen Jamie sind, erhalten möchte, kann man die WHERE-Klausel mit dem Gleichheitsoperator (=) wie folgt verwenden:

```
1 SELECT last_name, first_name  
2 FROM customer  
3 WHERE first_name = 'Jamie';
```

Wenn man den Kunden auswählen möchte, dessen Vorname Jamie und Nachname Reis lautet, kann man den logischen Operator AND verwenden, der zwei Bedingungen als folgende Abfrage kombiniert:

```
1 SELECT last_name, first_name  
2 FROM customer  
3 last_name='Rice';
```

Wenn man wissen möchte, wer die Leihgebühr mit einem Betrag von weniger als 1USD oder mehr als 8USD bezahlt hat, kann man die folgende Abfrage mit OR-Operator verwenden:

```
1 SELECT customer_id,amount,payment_date  
2 FROM payment  
3 WHERE amount <= 1 OR amount >= 8;
```