

5. SQL 問い合わせの基本

SELECT、FROM、WHERE、DISTINCT、IN、
BETWEEN、SQL 問い合わせの基本構造、
SQL による集計（AVG, SUM, COUNTなど）

URL: <https://www.kkaneko.jp/de/ds/index.html>

金子邦彦



謝辞：この資料では「いらすとや」のイラストを使用しています

アドバイス

- アクティブな学習を実践しよう

SQLを学ぶ際に、**プログラムを変更した結果を実際に見る**ことも心がけましょう。実際のデータベース操作を通じて学習を深めます。

- 簡単なスタートから

初めはシンプルなものからスタートしましょう。反復練習しましょう。

- ステップ・バイ・ステップで応用に進む

SQLスキルを向上させるために、**少しずつ難易度を上げ**、今まで自分ができなかったことにも**チャレンジ**しましょう。



- ① テーブルによるデータ管理の理解
- ② SQLの柔軟性の理解
- ③ SQLによるデータアクセスのスキル向上



アウトライン

1. イントロダクション
2. SELECT, FROM, WHERE の役割
3. WHERE による選択、IN
4. 重複の排除、集約
5. 演習

SQLFiddle のサイトにアクセス

Webブラウザを使用

1. ウェブブラウザを開く
2. アドレスバーにSQLFiddleのURLを入力

`http://sqlfiddle.com/`

URLが分からないときは、Googleなどの**検索エンジン**を利用。
「**SQLFiddle**」と**検索**し、表示された結果からSQLFiddleの
ウェブサイトをクリック。

SQLFiddle の画面

•左側のパネル: テーブル定義、データの追加など。SQLのCREATE TABLE や INSERT INTO などを入力。

•右側のパネル: SQL問い合わせ。SELECT などを入力。

The screenshot shows the SQLFiddle interface with a header bar that says "SQL Fiddle MySQL 5.6". Below the header, there are two panels. The left panel contains SQL code for creating a table and inserting data. The right panel contains a SELECT query. Below each panel are buttons for "Build Schema", "Edit Fullscreen", and "Run SQL". A red box highlights the "Run SQL" button in the right panel. Below the buttons, there is a table showing the results of the query.

```
1 CREATE TABLE employees (  
2   id INTEGER,  
3   name TEXT,  
4   age INTEGER,  
5   salary INTEGER,  
6   department_id INT);  
7 INSERT INTO employees VALUES (1, 'Alice', 30, 50000, 1);  
8 INSERT INTO employees VALUES (2, 'Bob', 40, 60000, 1);  
9 INSERT INTO employees VALUES (3, 'Charlie', 35, 70000, 2);
```

```
1 SELECT * FROM employees;  
2
```

Build Schema Edit Fullscreen Browser Run SQL Edit Fullscreen [;]

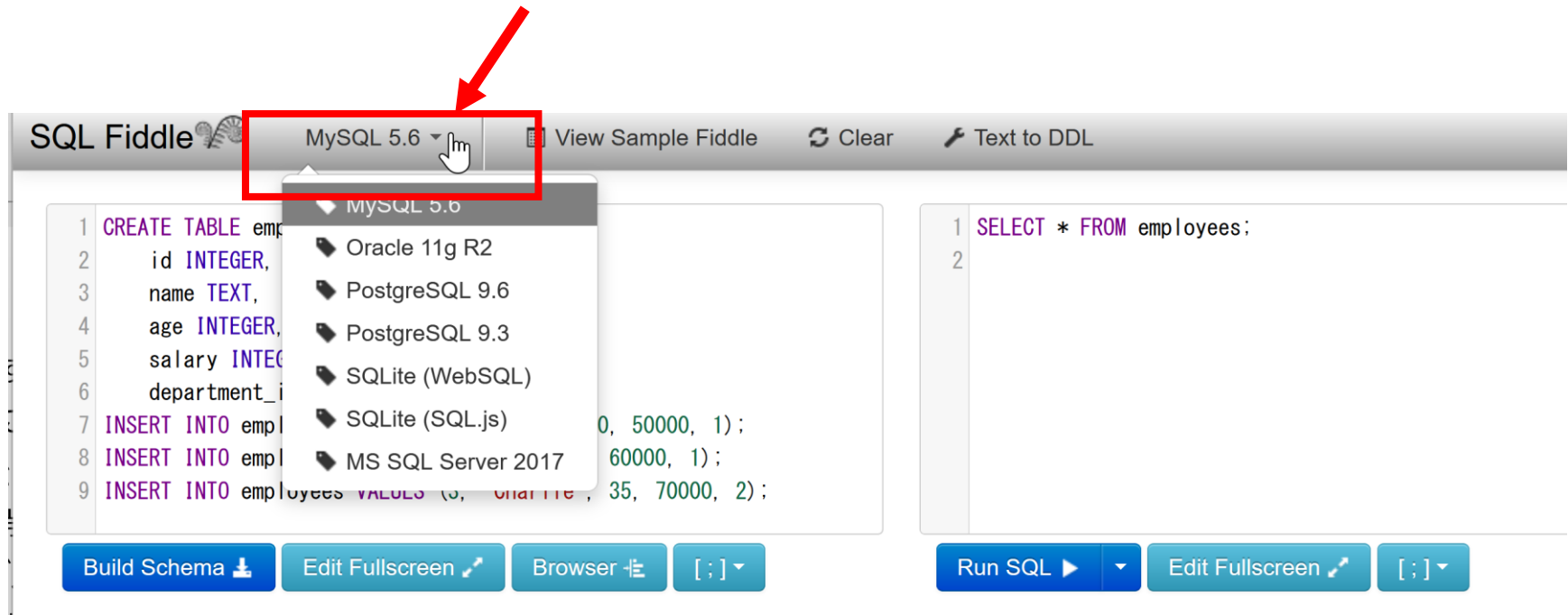
•実行ボタン

id	name	age	salary	department_id
1	Alice	30	50000	1
2	Bob	40	60000	1
3	Charlie	35	70000	2

結果ウィンドウ

SQLFiddle でのデータベース管理システムの選択 (高度な機能)

データベース管理システムの選択
(この授業では MySQL を使用)



5-1. イントロダクション

リレーショナルデータベースの仕組み

- データを**テーブル**と呼ばれる**表形式**で保存
- **テーブル間**は**関連**で結ばれる。複雑な構造を持ったデータを効率的に管理することを可能に。

ID	商品名	単価
1	みかん	50
2	りんご	100
3	メロン	500

関連

ID	購入者	商品ID	数量
1	X	1	10
2	Y	2	5

リレーショナルデータベースの重要性

1. **データの整合性:** リレーショナルデータベースは、**データの整合性を保持するための機能**を有する。これにより、誤ったデータや矛盾したデータが保存されるのを防ぐことができる。
2. **柔軟な問い合わせ（クエリ）能力:** リレーショナルデータベースの**SQL**（Structured Query Language）の使用により、**複雑な検索やデータの抽出**が可能になる。
3. **トランザクションの機能:** 一連の操作全体を一つの単位として取り扱うことができる機能。これにより、**データの一貫性と信頼性が向上**する。
4. **セキュリティ:** **アクセス権限の設定**などにより、セキュリティを確保。

データの安全な保管、効率的なデータ検索・操作、ビジネスや研究の意思決定をサポート。

SQL によるテーブル定義

- テーブル名 : **記録**
- 属性名 : **名前、得点、居室**
- 属性のデータ型 : **テキスト、数値、テキスト**
- データの整合性を保つための制約 : **なし**

```
CREATE TABLE 記録 (  
    名前 TEXT ,  
    得点 INTEGER ,  
    居室 TEXT ) ;
```

データ追加のSQL

記録

名前	得点	居室
徳川家康	85	1階
源義経	78	2階
西郷隆盛	90	3階
豊臣秀吉	82	1階
織田信長	75	2階

```
INSERT INTO 記録 VALUES ('徳川家康', 85, '1階');
```

```
INSERT INTO 記録 VALUES ('源義経', 78, '2階');
```

```
INSERT INTO 記録 VALUES ('西郷隆盛', 90, '3階');
```

```
INSERT INTO 記録 VALUES ('豊臣秀吉', 82, '1階');
```

```
INSERT INTO 記録 VALUES ('織田信長', 75, '2階');
```



演習 1. テーブル定義とデータの追加

【トピックス】

1. SQL によるテーブル定義
2. SQL によるデータの追加
3. 問い合わせ（クエリ）による確認

Webブラウザを使用

① アドレスバーにSQLFiddleのURLを入力

<http://sqlfiddle.com/>

URLが分からないときは、Googleなどの**検索エンジン**を利用。「**SQLFiddle**」と**検索**し、表示された結果からSQLFiddleのウェブサイトをクリック。



② 左側のパネルに、テーブル定義とデータの追加を行う SQL を入れる。

```
CREATE TABLE 記録 (  
    名前 TEXT,  
    得点 INTEGER,  
    居室 TEXT);  
  
INSERT INTO 記録 VALUES ('徳川家康', 85, '1階');  
INSERT INTO 記録 VALUES ('源義経', 78, '2階');  
INSERT INTO 記録 VALUES ('西郷隆盛', 90, '3階');  
INSERT INTO 記録 VALUES ('豊臣秀吉', 82, '1階');  
INSERT INTO 記録 VALUES ('織田信長', 75, '2階');
```

③ 「Build Schema」をクリック

SQL Fiddle  MySQL 5.6



```
1 CREATE TABLE 記録 (  
2     名前 TEXT,  
3     得点 INTEGER,  
4     居室 TEXT);  
5 INSERT INTO 記録 VALUES('徳川家康', 85, '1階');  
6 INSERT INTO 記録 VALUES('源義経', 78, '2階');  
7 INSERT INTO 記録 VALUES('西郷隆盛', 90, '3階');  
8 INSERT INTO 記録 VALUES('豊臣秀吉', 82, '1階');  
9 INSERT INTO 記録 VALUES('織田信長', 75, '2階');
```

Build Schema 

Edit Fullscreen 

Browser 

[;] 

Run SQL 

Edit Fullscreen 

[;] 

Please build schema.

④ 右側のパネルに、問い合わせ（クエリ）を行う SQL を入れる。

```
select * from 記録;
```

⑤ 「Run SQL」をクリック

SQL 文が**実行**され、結果が表示される。

⑥ 下側のウィンドウで、**結果を確認**。

The screenshot shows a database management interface. On the left, a SQL editor contains a script to create a table named '記録' (Record) with columns '名前' (Name), '得点' (Score), and '居室' (Room), followed by five INSERT statements. On the right, a query editor contains the SQL statement 'select * from 記録;'. Below the editors, a 'Run SQL' button is highlighted with a red box. At the bottom, a table displays the results of the query, which are the five rows inserted by the script. This results table is also highlighted with a red box.

名前	得点	居室
徳川家康	85	1階
源義経	78	2階
西郷隆盛	90	3階
豊臣秀吉	82	1階
織田信長	75	2階

あとで使用するのでブラウザを閉じないこと

自習

次の SQL を試してみる。

SELECT 名前 FROM 記録;

SELECT 得点 FROM 記録;

結果

名前
徳川家康
源義経
西郷隆盛
豊臣秀吉
織田信長

✓ Record Count: 5; Execution Time: 4ms [+ View Execution Plan](#)

得点
85
78
90
82
75

✓ Record Count: 5; Execution Time: 1ms [+ View Execution Plan](#)

あとで使用するのでブラウザを閉じないこと

5-2. SELECT、FROM、WHERE の役割

SELECT, FROM, WHERE を学ぶことのメリット

- **テーブルによるデータ管理**

リレーショナルデータベースのテーブルによるデータ管理についての理解が深まる。

- **SQLの柔軟性**

条件を指定してテーブルからデータを取得する多彩な方法を学ぶ。

- **SQL によるデータアクセス**

テーブルから必要なデータを取得するスキルを習得。

SQL の select, from, where

select

問い合わせ（クエリ）のための基本的な命令。

取得したいデータの指定

from

データ取得の対象となるテーブルを指定

例： **select * from** テーブル名;

where

特定の条件を満たす行の選択

例： **select * from** テーブル名 **where** 列1 = 値1;

SQL 理解のための前提知識

○ テーブル

データを**テーブル**と呼ばれる**表形式**で保存

名前	得点	居室
徳川家康	85	1階
源義経	78	2階
西郷隆盛	90	3階
豊臣秀吉	82	1階
織田信長	75	2階

○ 問い合わせ（クエリ）

- **問い合わせ（クエリ）**は、**データベース**から必要なデータを検索、加工するための指令
- SELECT, FROM, WHERE など、**多様**なコマンドが存在。
- **結合、集計、ソート、副問い合わせ**など、高度な操作も可能

テーブル「記録」からデータを取得するSQLの例

- **SELECT * FROM 記録;**
- **SELECT 居室 FROM 記録;**
- **SELECT DISTINCT 居室 FROM 記録;**
- **SELECT 名前, 得点 FROM 記録 WHERE 得点 > 80;**
- **SELECT 名前, 得点 FROM 記録 WHERE 得点 BETWEEN 80 AND 85;**
- **SELECT AVG(得点) FROM 記録;**

AVG, MAX, MIN, SUM: 平均、最大、最小、合計

COUNT: 行数

- **SELECT * FROM 記録 WHERE 居室 LIKE '%階';**
「*階」、「階*」、「*階*」のように書くことができる
Access では % でなく *
- **SELECT * FROM 記録 WHERE 居室 IN ('1階', '2階');**

5-3. WHERE による選択、IN の役割

WHERE による選択

問い合わせ（クエリ）では、**条件を指定**することにより、**データの行単位での選択**を行う

WHERE 得点 > 80

「**得点が80より大きい**」行を選択

WHERE 得点 **BETWEEN** 80 **AND** 85

「**得点が80以上かつ85以下**」の範囲にある行を選択

WHERE 居室 **LIKE** '%階';

「**居室が'階'で終わる**」行を選択

ワイルドカード文字（*）は任意の文字列を表す

WHERE 居室 **IN** ('1階', '2階');

「**居室が'1階'または'2階'**」のいずれかに一致する行を選択

IN の役割

複数の値と比較.

そのうち1つの値でも一致するものを結果とする

「居室が'1階'または'2階」のいずれかに一致する行を選択

```
SELECT *  
FROM 記録  
WHERE 居室 IN ('1階', '2階');
```

半角丸かっこ
で囲む

半角の
カンマ

半角丸かっこ
で囲む

5-4. 重複行の除去、集約

DISTINCT は重複行の除去

SELECT 居室 FROM 記録;

結果

居室
1階
2階
3階
1階
2階

SELECT DISTINCT 居室 FROM 記録

結果

居室
1階
2階
3階

集約

AVG, MAX, MIN, SUM: 平均、最大、最小、合計

COUNT: 行数

記録

名前	得点	居室
徳川家康	85	1階
源義経	78	2階
西郷隆盛	90	3階
豊臣秀吉	82	1階
織田信長	75	2階

SELECT AVG(得点) FROM 記録;
82


SELECT MAX(得点) FROM 記録;
90

SELECT MIN(得点) FROM 記録;
75

SELECT SUM(得点) FROM 記録;
410

SELECT * FROM 記録;	テーブルのすべて
SELECT 居室 FROM 記録;	「居室」の列をすべて
SELECT DISTINCT 居室 FROM 記録;	重複行（同じ値の行）を除去
SELECT 名前, 得点 FROM 記録 WHERE 得点 > 80;	「名前」と「得点」の列で、「得点が80より大きい」行を選択
SELECT 名前, 得点 FROM 記録 WHERE 得点 BETWEEN 80 AND 85;	「名前」と「得点」の列で、「得点が80以上かつ85以下」の範囲にある行を選択
SELECT AVG (得点) FROM 記録;	すべての「得点」の値の平均
SELECT * FROM 記録 WHERE 居室 LIKE '%階';	「居室が'階'で終わる」行を選択
SELECT * FROM 記録 WHERE 居室 IN ('1階', '2階')	「居室が'1階'または'2階」のいずれかに一致する行を選択

5-5. 演習



演習 2. SQL による問い合わせ (クエリ)

【トピックス】

1. 選択
2. DISTINCT による重複行除去
3. パターンマッチング
4. IN を用いた条件指定

① 右側のパネルに、問い合わせ（クエリ）を行う SQL を入れる。

```
SELECT 居室 FROM 記録;  
SELECT DISTINCT 居室 FROM 記録;
```

② 「Run SQL」をクリック
SQL 文が**実行**され、結果が表示される。

③ 下側のウィンドウで、**結果を確認**。

The screenshot shows a SQL query editor interface. The left panel contains the following SQL code:

```
1 CREATE TABLE 記録 (  
2   名前 TEXT,  
3   得点 INTEGER,  
4   居室 TEXT);  
5 INSERT INTO 記録 VALUES('徳川家康', 85, '1階');  
6 INSERT INTO 記録 VALUES('源義経', 78, '2階');  
7 INSERT INTO 記録 VALUES('西郷隆盛', 90, '3階');  
8 INSERT INTO 記録 VALUES('豊臣秀吉', 82, '1階');  
9 INSERT INTO 記録 VALUES('織田信長', 75, '2階');  
10  
11
```

The right panel shows the SQL queries entered:

```
SELECT 居室 FROM 記録;  
SELECT DISTINCT 居室 FROM 記録;
```

The 'Run SQL' button is highlighted with a red box. Below the editor, there are two result tables, both highlighted with red boxes:

居室
1階
2階
3階
1階
2階

Record Count: 5; Execution Time: 15ms + View Execution Plan + link

居室
1階
2階
3階

④ 右側のパネルに、問い合わせ（クエリ）を行う SQL を入れる。（以前の SQL は不要なので消す）

```
SELECT 名前, 得点 FROM 記録 WHERE 得点 > 80;  
SELECT 名前, 得点 FROM 記録 WHERE 得点 BETWEEN 80 AND 85;
```

⑤ 「Run SQL」をクリック

SQL 文が**実行**され、結果が表示される。

⑥ 下側のウィンドウで、**結果を確認**。

The screenshot shows a SQL execution interface. On the left, a code editor contains two SQL queries. The first query is highlighted with a red box: `SELECT 名前, 得点 FROM 記録 WHERE 得点 > 80;`. The second query is `SELECT 名前, 得点 FROM 記録 WHERE 得点 BETWEEN 80 AND 85;`. Below the code editor, there are buttons for 'Build Schema', 'Edit Fullscreen', 'Browser', and a dropdown menu showing 'Executing SQL...'. On the right, there are two result tables. The first table, also highlighted with a red box, shows the results of the first query: 3 records with names 徳川家康, 西郷隆盛, and 豊臣秀吉. The second table shows the results of the second query: 2 records with names 徳川家康 and 豊臣秀吉. Below each table, there is a status bar showing the record count and execution time.

```
1 CREATE TABLE 記録 (  
2   名前 TEXT,  
3   得点 INTEGER,  
4   居室 TEXT);  
5 INSERT INTO 記録 VALUES ('徳川家康', 85, '1階');  
6 INSERT INTO 記録 VALUES ('源義経', 78, '2階');  
7 INSERT INTO 記録 VALUES ('西郷隆盛', 90, '3階');  
8 INSERT INTO 記録 VALUES ('豊臣秀吉', 82, '1階');  
9 INSERT INTO 記録 VALUES ('織田信長', 75, '2階');  
10
```

SELECT 名前, 得点 FROM 記録 WHERE 得点 > 80;
SELECT 名前, 得点 FROM 記録 WHERE 得点 BETWEEN 80 AND 85;

Executing SQL...

名前	得点
徳川家康	85
西郷隆盛	90
豊臣秀吉	82

Record Count: 3; Execution Time: 40ms + View Execution Plan link

名前	得点
徳川家康	85
豊臣秀吉	82

Record Count: 2; Execution Time: 18ms + View Execution Plan link

⑦ 右側のパネルに、問い合わせ（クエリ）を行う SQL を入れる。（以前の SQL は不要なので消す）

```
SELECT AVG(得点) FROM 記録;
```

⑧ 「Run SQL」をクリック

SQL 文が**実行**され、結果が表示される。

⑨ 下側のウィンドウで、**結果を確認**。

```
1 CREATE TABLE 記録 (  
2   名前 TEXT,  
3   得点 INTEGER,  
4   居室 TEXT);  
5 INSERT INTO 記録 VALUES('徳川家康', 85, '1階');  
6 INSERT INTO 記録 VALUES('源義経', 78, '2階');  
7 INSERT INTO 記録 VALUES('西郷隆盛', 90, '3階');  
8 INSERT INTO 記録 VALUES('豊臣秀吉', 82, '1階');  
9 INSERT INTO 記録 VALUES('織田信長', 75, '2階');
```

```
1 SELECT AVG(得点) FROM 記録;
```

AVG(得点)
82

Record Count: 1; Execution Time: 4ms + View Execution Plan link

⑩ 右側のパネルに、問い合わせ（クエリ）を行う SQL を入れる。（以前の SQL は不要なので消す）

```
SELECT * FROM 記録 WHERE 居室 LIKE '%階';  
SELECT * FROM 記録 WHERE 居室 IN ('1階', '2階');
```

⑪ 「Run SQL」をクリック

SQL 文が**実行**され、結果が表示される。

⑫ 下側のウィンドウで、**結果を確認**。

The screenshot shows a SQL IDE interface. The top panel contains two SQL queries. The first query is `SELECT * FROM 記録 WHERE 居室 LIKE '%階';` and the second query is `SELECT * FROM 記録 WHERE 居室 IN ('1階', '2階');`. The 'Run SQL' button is highlighted. Below the queries, there are two result tables. The first table shows the results of the first query, and the second table shows the results of the second query. Both tables have columns for '名前' (Name), '得点' (Score), and '居室' (Room).

名前	得点	居室
徳川家康	85	1階
源義経	78	2階
西郷隆盛	90	3階
豊臣秀吉	82	1階
織田信長	75	2階

Record Count: 5; Execution Time: 13ms [View Execution Plan](#) [link](#)

名前	得点	居室
徳川家康	85	1階
源義経	78	2階
豊臣秀吉	82	1階
織田信長	75	2階

Record Count: 4; Execution Time: 4ms [View Execution Plan](#) [link](#)

自習

- テーブル「記録」を使用してください
- 後ろのページに解答例を載せているので活用してください
- SQLFiddle などを利用し、実際に実行してみると、理解が進み、自分で間違いに気づくこともでき効果的です。

自習 1. 得点が80以上の人物を選択する

目的: WHERE句を使用して特定の条件を満たす行を選択する方法を学ぶ

得点が80以上のすべての人物を選択してください。

ヒント: WHERE句を使用して得点の条件を指定

自習 2. 1階に住む人物の名前を重複なく選択する

目的: DISTINCTを使用して重複する値を排除する方法を学ぶ。

1階に住む人物の名前を選択してください。そのとき、重複なく表示するようにしてください。

ヒント: DISTINCTとWHERE句を組み合わせて使用

自習 3. 得点が70から90の範囲内の人物の名前と居室を選択

目的: BETWEENを使用して、特定の範囲内のデータを選択する方法を学ぶ。

BETWEEN を用いて、得点が70から90の範囲内のすべての人物の名前と居室を選択してください。得点が 70, 90 の人も含めてください。

ヒント: BETWEEN を使用して得点の範囲を指定

自習 4 . '織田'で始まる名前の人を選択

目的: LIKEを使用して特定のパターンに一致する行を選択する方法を学ぶ。

'織田'で始まる名前の人をすべて選択してください。

ヒント: LIKEとワイルドカード%を使用してパターンマッチングを行います。

自習 5. 得点の最大値を得る

目的: MAX を使用して「得点」の列の最大値を得る方法を学ぶ

得点の最大値を得てください

ヒント: SELECT MAX(得点) のように書くことで、得点の最大値を得ることができます。

自習 1 **解答例:** SELECT * FROM 記録 WHERE 得点 >= 80;

自習 2 **解答例:** SELECT DISTINCT 名前 FROM 記録
WHERE 居室 = '1階';

自習 3 **解答例:** SELECT 名前, 居室 FROM 記録 WHERE 得点 BETWEEN 70 AND 90;

自習 4 **解答例 :** SELECT * FROM 記録 WHERE 名前 LIKE '織田%';

自習 5 **解答例 :** SELECT MAX(*) FROM;

全体まとめ ①

SQL理解の前提知識

- **テーブル**: データを表形式で保存する構造。
- **問い合わせ（クエリ）**: データベースからデータを検索・加工するための指令。SELECT、FROM、WHEREなどのコマンドが存在し、高度な操作も可能。

SQLの select, from, where

select

- 問い合わせ（クエリ）のための基本的な命令。
- 取得したいデータの指定

from

- データ取得の対象となるテーブルを指定

where

- 特定の条件を満たす行の選択

全体まとめ ②

WHERE による選択

問い合わせ（クエリ）では、条件を指定してデータの行単位で選択。

- WHERE 得点 > 80 **得点が80より大きい行**を選択。
- WHERE 得点 BETWEEN 80 AND 85 **得点が80以上かつ85以下の行**を選択。
- WHERE 居室 LIKE '%階'; **居室が'階'で終わる行**を選択。ワイルドカード（%）は任意の文字列を表す。
- WHERE 居室 IN ('1階', '2階') **居室が'1階'または'2階'に一致する行**を選択。

全体まとめ ③

DISTINCTは重複行の除去

- SELECT 居室 FROM 記録
- SELECT DISTINCT 居室 FROM 記録

居室
1階
2階
3階
1階
2階

居室
1階
2階
3階

集約

AVG, MAX, MIN, SUM: 平均、最大、最小、合計。

- SELECT AVG(得点) FROM 記録;
- SELECT MAX(得点) FROM 記録;
- SELECT MIN(得点) FROM 記録;
- SELECT SUM(得点) FROM 記録;



① テーブルによるデータ管理の理解

SELECT、FROM、WHEREを学ぶことで、リレーショナルデータベース内のテーブルにデータをどのように管理するかについて深い理解が得られます。

② SQLの柔軟性の理解

SQLは柔軟な言語です。SQLの柔軟性を学ぶことで、必要なデータを効率的に取得できるスキルを習得できます。

③ SQLによるデータアクセスのスキル向上

SELECT、FROM、WHEREを理解し、正確に使用できるようになることで、データベースから必要なデータを取得するスキルが向上します。これはデータ分析、レポート作成、ビジネスの意思決定において有用です。