第7回：表をくっつける
- INNER JOIN -

IDで管理された2つの表を結合し、意味のある情報として
表示できるようになる

● 本日のゴール

✅ IDで管理された2つの表を結合し、意味のある情報として表示できるようになる

● 座学パート

1. なぜJOINが必要？

現実のデータベースでは、データを複数の表に分けて管理します。

例：売上データの問題

```
sales.csv：
customer_id | product_id | quantity
C001        | P003       | 5

→ 顧客名は？ customers.csvが必要
→ 商品名は？ products.csvが必要
```

2. INNER JOINの基本構文

```
SELECT 列名
FROM テーブル1
INNER JOIN テーブル2
ON テーブル1.共通列 = テーブル2.共通列;
```

3. JOINの仕組み

```
customers表:                    sales表:
customer_id | name              customer_id | product_id
C001        | 田中              C001        | P003
C002        | 佐藤              C002        | P001

結合結果:
customer_id | name | product_id
C001        | 田中 | P003
C002        | 佐藤 | P001
```

4. テーブルの別名（エイリアス）

長いテーブル名を短く書ける：

```
FROM 'data/sales.csv' AS s
INNER JOIN 'data/products.csv' AS p
ON s.product_id = p.product_id
```

● 演習パート

演習1：売上データに商品名を表示

```
SELECT
    s.*,
    p.product_name
FROM 'data/sales.csv' AS s
INNER JOIN 'data/products.csv' AS p
ON s.product_id = p.product_id;
```

結果を確認：

- product_idだけでなく、商品名が表示される

列を選んで見やすく

```
SELECT
    s.order_date AS 売上日,
    s.customer_id AS 顧客ID,
    p.product_name AS 商品名,
    s.quantity AS 数量
FROM 'data/sales.csv' AS s
INNER JOIN 'data/products.csv' AS p
ON s.product_id = p.product_id
ORDER BY s.order_date DESC;
```

演習2：売上データに顧客名を表示

```
SELECT
    s.order_date AS 売上日,
    c.customer_name AS 顧客名,
    s.product_id AS 商品ID,
    s.quantity AS 数量
FROM 'data/sales.csv' AS s
INNER JOIN 'data/customers.csv' AS c
ON s.customer_id = c.customer_id;
```

誰が何を買ったか分かりやすい！

演習3：【応用】価格を含めて売上金額を計算

```
SELECT
    s.order_date AS 売上日,
    s.customer_id AS 顧客ID,
    p.product_name AS 商品名,
    p.price AS 単価,
    s.quantity AS 数量,
    p.price * s.quantity AS 売上金額
FROM 'data/sales.csv' AS s
INNER JOIN 'data/products.csv' AS p
ON s.product_id = p.product_id
ORDER BY 売上金額 DESC;
```

計算もできる！売上金額の大きい順に表示。

応用練習

1. 3つのテーブルを結合

```
-- 売上に顧客名と商品名の両方を表示
SELECT
    s.order_date AS 売上日,
    c.customer_name AS 顧客名,
    p.product_name AS 商品名,
    s.quantity AS 数量,
    p.price * s.quantity AS 売上金額
FROM 'data/sales.csv' AS s
INNER JOIN 'data/customers.csv' AS c ON s.customer_id = c.customer_id
INNER JOIN 'data/products.csv' AS p ON s.product_id = p.product_id
ORDER BY s.order_date DESC;
```

2. 条件を加えた結合

```
-- 特定顧客の購入履歴を商品名付きで
SELECT
    c.customer_name AS 顧客名,
    p.product_name AS 商品名,
    s.quantity AS 数量,
    s.order_date AS 購入日
FROM 'data/sales.csv' AS s
INNER JOIN 'data/customers.csv' AS c ON s.customer_id = c.customer_id
INNER JOIN 'data/products.csv' AS p ON s.product_id = p.product_id
WHERE c.customer_name = '田中太郎'
ORDER BY s.order_date;
```

3. 集計と組み合わせ

```
-- 顧客ごとの購入金額合計
SELECT
    c.customer_name AS 顧客名,
    SUM(p.price * s.quantity) AS 購入金額合計
FROM 'data/sales.csv' AS s
INNER JOIN 'data/customers.csv' AS c ON s.customer_id = c.customer_id
INNER JOIN 'data/products.csv' AS p ON s.product_id = p.product_id
GROUP BY c.customer_id, c.customer_name
ORDER BY 購入金額合計 DESC;
```

実践的な使い方

ケース1：売上レポート

```
SELECT -- 見やすい売上明細
    s.order_date AS 売上日,
    c.customer_name AS 顧客名,
    c.email AS メール,
    p.product_name AS 商品名,
    p.category AS カテゴリ,
    s.quantity AS 数量,
    p.price AS 単価,
    p.price * s.quantity AS 金額
FROM 'data/sales.csv' AS s
INNER JOIN 'data/customers.csv' AS c ON s.customer_id = c.customer_id
INNER JOIN 'data/products.csv' AS p ON s.product_id = p.product_id
WHERE s.order_date >= '2024-01-20'
ORDER BY s.order_date DESC;
```

ケース2：商品別売上分析

```
-- 商品別の売上統計
SELECT
    p.product_name AS 商品名,
    p.category AS カテゴリ,
    COUNT(*) AS 販売回数,
    SUM(s.quantity) AS 総販売数,
    SUM(p.price * s.quantity) AS 売上金額
FROM 'data/sales.csv' AS s
INNER JOIN 'data/products.csv' AS p ON s.product_id = p.product_id
GROUP BY p.product_id, p.product_name, p.category
ORDER BY 売上金額 DESC;
```

JOINを理解する

段階的に確認

1. まず個別のテーブルを確認

```
-- sales.csvの内容
SELECT * FROM 'data/sales.csv' LIMIT 3;

-- products.csvの内容
SELECT * FROM 'data/products.csv' LIMIT 3;
```

2. JOINした結果を確認

```
SELECT s.*, p.product_name
FROM 'data/sales.csv' AS s
INNER JOIN 'data/products.csv' AS p
ON s.product_id = p.product_id
LIMIT 3;
```

→ 結合前後の違いが視覚的に分かる！

よくあるエラーと対処法

⚠️ **列名の曖昧さエラー**：

```
-- customer_idがどちらのテーブルか不明
SELECT customer_id  -- エラー！
FROM 'data/sales.csv' AS s
INNER JOIN 'data/customers.csv' AS c
ON s.customer_id = c.customer_id;
```

💡 **テーブル名を明示：**

```
SELECT s.customer_id  -- OK!
FROM 'data/sales.csv' AS s
INNER JOIN 'data/customers.csv' AS c
ON s.customer_id = c.customer_id;
```

JOIN使用時のTips

1. 結合キーの確認

結合する前に、共通の列（キー）を確認：

- sales.csv ← customer_id → customers.csv
- sales.csv ← product_id → products.csv

2. パフォーマンスの考慮

- 必要な列だけSELECT
- WHERE句で早めに絞り込む
- インデックスがあれば高速（今回は関係なし）

3. 結合の種類

- **INNER JOIN**：両方に存在するデータのみ（今回学習）
- **LEFT JOIN**：左側の全データ（次回学習）
- **RIGHT JOIN**：右側の全データ
- **FULL OUTER JOIN**：両方の全データ

● 本日のまとめ

今日学んだこと：

- ✅ `INNER JOIN` で複数の表を結合
- ✅ `ON` 句で結合条件を指定
- ✅ IDだけでなく意味のある情報を表示
- ✅ 複数のJOINも可能
- ✅ テーブル名.列名でテーブルを明示

よく使うパターン

```sql
-- 2テーブル結合
SELECT
    t1.列名,
    t2.列名
FROM テーブル1 AS t1
INNER JOIN テーブル2 AS t2
ON t1.共通列 = t2.共通列;

-- 3テーブル結合
FROM テーブル1 AS t1
INNER JOIN テーブル2 AS t2 ON t1.id = t2.id
INNER JOIN テーブル3 AS t3 ON t2.id = t3.id;
```

次回予告

第8回では、LEFT JOINを学び、これまでの知識を総動員した総合演習を行います。
購入履歴のない顧客を見つけたり、複雑な分析にチャレンジ！

● 追加演習問題

問題1：基本的なJOIN

以下の結合を実行：

```
-- 1. salesとcustomersを結合し、顧客名付きの売上一覧を表示
-- あなたの答えをここに書いてください

-- 2. salesとproductsを結合し、商品名と価格付きの売上一覧を表示
-- あなたの答えをここに書いてください

-- 3. productsとsalesを結合し、売れた商品の情報だけを表示
-- あなたの答えをここに書いてください
```

問題2：JOINした結果の加工

結合後にデータを整形：

```
-- salesとproductsを結合して、以下の形式で表示：
-- 売上日 | 商品名 | 単価 | 数量 | 小計（単価×数量）
SELECT
    s.order_date AS 売上日,
    ___ AS 商品名,
    ___ AS 単価,
    s.quantity AS 数量,
    ___ AS 小計
FROM 'data/sales.csv' AS s
INNER JOIN 'data/products.csv' AS p ON ___;
```

問題3：3つのテーブルを結合

全ての情報を1つにまとめる：

```
-- sales、customers、productsを全て結合
-- 顧客名、商品名、カテゴリ、数量、金額を表示
-- あなたの答えをここに書いてください
```

問題4：条件付きJOIN

WHERE句と組み合わせ：

```
-- 1. 電子機器カテゴリの売上だけを、顧客名付きで表示
-- あなたの答えをここに書いてください

-- 2. 特定期間（1月20日以降）の売上を、商品名・顧客名付きで表示
-- あなたの答えをここに書いてください

-- 3. 高額商品（3万円以上）の購入履歴を、顧客情報付きで表示
-- あなたの答えをここに書いてください
```

🎯 チャレンジ問題

```
-- 「どの顧客がどのカテゴリの商品をいくつ買ったか」を集計
-- ヒント：JOINとGROUP BYを組み合わせる
```

💡 実践問題：売上レポート

```
-- 以下のレポートを作成：
-- 1. 日別売上明細（顧客名、商品名、金額付き）
-- 2. 顧客別購入商品リスト（重複なし）
-- 3. カテゴリ別売上集計（顧客数、総数量、総金額）
```

🔍 デバッグ練習

以下のクエリの問題を修正：

```
SELECT customer_name, product_name
FROM 'data/sales.csv' AS s
INNER JOIN 'data/customers.csv' AS c ON customer_id = customer_id;

SELECT * FROM 'data/sales.csv' s
INNER JOIN 'data/products.csv' p ON s.customer_id = p.product_id;

SELECT s.sale_id, c.name, p.item_name
FROM 'data/sales.csv' s
INNER JOIN 'data/customers.csv' c ON s.customer_id = c.customer_id
INNER JOIN 'data/products.csv' p ON s.product_id = p.product_id;
```

📊 応用：JOINの活用

```sql
-- 1. 自己結合の概念（同じ顧客の他の購入を探す）
-- 例：C001が買った商品を、他に誰が買ったか
SELECT DISTINCT s2.customer_id, c.customer_name
FROM 'data/sales.csv' s1
INNER JOIN 'data/sales.csv' s2 ON s1.product_id = s2.product_id
INNER JOIN 'data/customers.csv' c ON s2.customer_id = c.customer_id
WHERE s1.customer_id = 'C001' AND s2.customer_id != 'C001';

-- 2. 複雑な集計
-- 顧客ごとの「お気に入りカテゴリ」を見つける
```

🎮 総合演習：360度顧客ビュー

```
-- 特定顧客（C001）の完全な購買履歴を作成：
-- 購入日、商品名、カテゴリ、単価、数量、小計
-- さらに以下も追加：
--  - その顧客の総購入金額
--  - 平均購入単価
--  - よく買うカテゴリTOP3
```

🏆 上級問題：クロスセル分析

```
-- 「この商品を買った人は、こんな商品も買っています」分析
-- 特定商品（P001）を買った顧客が、他に何を買っているか
-- 商品名付きで、購入人数の多い順に表示
```

📈 パフォーマンス最適化

```sql
-- 以下の2つのクエリの違いを理解：
-- パターン1：WHERE句で早めに絞り込み
SELECT c.customer_name, p.product_name, s.quantity
FROM 'data/sales.csv' s
INNER JOIN 'data/customers.csv' c ON s.customer_id = c.customer_id
INNER JOIN 'data/products.csv' p ON s.product_id = p.product_id
WHERE s.order_date >= '2024-01-20';

-- パターン2：全部結合してから絞り込み（非効率）
--  （実際は同じですが、考え方として）
```

FAQ

Q: JOINとINNER JOINの違いは？

A: 同じです。JOINと書くとINNER JOINになります。

Q: 結合できない（0件になる）

A: ON句の条件を確認。データ型や値が一致しているか要チェック。

Q: どのテーブルから書けばいい？

A: 一般的には、メインとなるテーブル（今回ならsales）から書きます。