

# 第8回：総合演習と LEFT JOIN

---

これまでの知識を組み合わせ、実践的なデータ抽出に挑戦する

# ● 本日のゴール

---

- ✓ これまでの知識を組み合わせ、実践的なデータ抽出に挑戦する
- ✓ LEFT JOINで「存在しないデータ」も見つけられるようになる

# ● 座学パート

---

## 1. これまでの復習

**SELECT** : 必要な列を選ぶ

**WHERE** : 条件で行を絞る

**ORDER BY** : 並び替える

**COUNT/SUM/AVG** : 集計する

**GROUP BY** : グループ化して集計

**INNER JOIN** : 共通データを結合

## 2. LEFT JOINとは？

INNER JOINとの違い：

**INNER JOIN**：両方のテーブルに存在するデータのみ

**LEFT JOIN**：左側のテーブルの全データ（右側になくてもOK）

```
-- 例：購入履歴のない顧客も表示
FROM customers AS c
LEFT JOIN sales AS s
ON c.customer_id = s.customer_id
```

## 3. LEFT JOINの使いどころ

購入履歴のない顧客を探す

在庫のない商品を見つける

売れていない商品を特定する

## 4. NULLの扱い

LEFT JOINで右側にデータがない場合、その列はNULLになる：

customers表：

C001		田中
C002		佐藤
C003		鈴木

sales表：

C001		P003
C001		P002

(C003の売上なし)

LEFT JOIN結果：

C001		田中		P003
C001		田中		P002
C002		佐藤		NULL ← 売上がないのでNULL

# ● 演習パート

---

## 演習1：購入履歴のない顧客を探す

```
SELECT
    c.customer_id AS 顧客ID,
    c.customer_name AS 顧客名,
    c.email AS メール,
    s.order_date AS 最終購入日
FROM 'data/customers.csv' AS c
LEFT JOIN 'data/sales.csv' AS s
ON c.customer_id = s.customer_id
WHERE s.customer_id IS NULL;
```

結果：C006（伊藤次郎）とC007（渡辺美香）が表示される！

## なぜIS NULLで判定？

LEFT JOINで右側（sales）にデータがない場合、その列がNULLになるため。



## 演習2：【最終課題】 顧客ごとの購入金額ランキング

これまでの知識を総動員！

```
SELECT
    c.customer_id AS 顧客ID,
    c.customer_name AS 顧客名,
    COUNT(s.order_date) AS 購入回数,
    COALESCE(SUM(s.quantity * p.price), 0) AS 合計購入金額
FROM 'data/customers.csv' AS c
LEFT JOIN 'data/sales.csv' AS s ON c.customer_id = s.customer_id
LEFT JOIN 'data/products.csv' AS p ON s.product_id = p.product_id
GROUP BY c.customer_id, c.customer_name
ORDER BY 合計購入金額 DESC;
```

# ポイント

3つのテーブルを結合

LEFT JOINで購入履歴のない顧客も含める

GROUP BYで顧客ごとに集計

COALESCE関数でNULLを0に変換

ORDER BYでランキング表示

# ● 応用練習

---

## 1. 売れていない商品を探す

```
SELECT
    p.product_id AS 商品ID,
    p.product_name AS 商品名,
    p.price AS 価格,
    COUNT(s.product_id) AS 販売回数
FROM 'data/products.csv' AS p
LEFT JOIN 'data/sales.csv' AS s
ON p.product_id = s.product_id
GROUP BY p.product_id, p.product_name, p.price
HAVING COUNT(s.product_id) = 0;
```

## 2. 顧客別・商品別の購入マトリックス

```
SELECT
    c.customer_name AS 顧客名,
    p.product_name AS 商品名,
    COALESCE(SUM(s.quantity), 0) AS 購入数
FROM 'data/customers.csv' AS c
CROSS JOIN 'data/products.csv' AS p
LEFT JOIN 'data/sales.csv' AS s
    ON c.customer_id = s.customer_id
    AND p.product_id = s.product_id
GROUP BY c.customer_id, c.customer_name, p.product_id, p.product_name
ORDER BY c.customer_name, p.product_name;
```

### 3. 月次売上レポート（仮想）

-- 日付ごとの売上金額（商品・顧客情報付き）

**SELECT**

s.order\_date **AS** 売上日,  
c.customer\_name **AS** 顧客名,  
p.product\_name **AS** 商品名,  
p.category **AS** カテゴリ,  
s.quantity **AS** 数量,  
p.price **AS** 単価,  
s.quantity \* p.price **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**, 売上金額 **DESC**;

# ● 実践的な分析例

## ケース1：優良顧客分析

```
-- 購入金額上位の顧客とその詳細
WITH customer_summary AS (
  SELECT
    c.customer_id,
    c.customer_name,
    c.email,
    COUNT(DISTINCT s.order_date) AS 購入日数,
    COUNT(s.order_date) AS 購入回数,
    COALESCE(SUM(s.quantity * p.price), 0) AS 合計金額
  FROM 'data/customers.csv' AS c
  LEFT JOIN 'data/sales.csv' AS s ON c.customer_id = s.customer_id
  LEFT JOIN 'data/products.csv' AS p ON s.product_id = p.product_id
```

## ケース2：商品パフォーマンス分析

-- 商品別の売上パフォーマンス

**SELECT**

p.product\_name **AS** 商品名,

p.category **AS** カテゴリ,

p.price **AS** 単価,

**COUNT**(s.product\_id) **AS** 販売回数,

**COALESCE**(**SUM**(s.quantity), 0) **AS** 総販売数,

**COALESCE**(**SUM**(s.quantity \* p.price), 0) **AS** 売上金額,

**CASE**

**WHEN** **COUNT**(s.product\_id) = 0 **THEN** '未販売'

**WHEN** **SUM**(s.quantity \* p.price) > 100000 **THEN** '売れ筋'

**ELSE** '通常'

**END AS** 販売状況

**FROM** 'data/products.csv' **AS** p

**LEFT JOIN** 'data/sales.csv' **AS** s **ON** p.product\_id = s.product\_id

**GROUP BY** p.product\_id, p.product\_name, p.category, p.price

**ORDER BY** 売上金額 **DESC**;

# ● SQLマスターへの道

---

## 今後学ぶと良い構文

**WITH句 (CTE)** : 複雑なクエリを分かりやすく

**ウィンドウ関数** : 順位付けや累計計算

**CASE文** : 条件分岐

**サブクエリ** : クエリの中にクエリ

**UNION** : 複数の結果を結合



# 実務でのSQL活用

**レポート作成**：定期的な売上レポート

**データ分析**：顧客行動分析、商品分析

**異常検知**：通常と異なるパターンの発見

**ダッシュボード**：リアルタイムな状況把握

# ● 総合演習の解答例

---

## 基本レベル：各種集計

-- 顧客数、商品数、売上件数

**SELECT**

```
(SELECT COUNT(*) FROM 'data/customers.csv') AS 顧客数,  
(SELECT COUNT(*) FROM 'data/products.csv') AS 商品数,  
(SELECT COUNT(*) FROM 'data/sales.csv') AS 売上件数;
```

## 中級レベル：クロス集計

-- カテゴリ別・顧客別の購入金額

**SELECT**

c.customer\_name **AS** 顧客名,

p.category **AS** カテゴリ,

SUM(s.quantity \* p.price) **AS** 購入金額

**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

**GROUP BY** c.customer\_name, p.category

**ORDER BY** c.customer\_name, p.category;

# 上級レベル：複雑な分析

-- 顧客の購入傾向分析

**SELECT**

c.customer\_name **AS** 顧客名,  
COUNT(**DISTINCT** p.category) **AS** 購入カテゴリ数,  
COUNT(**DISTINCT** p.product\_id) **AS** 購入商品種類,  
STRING\_AGG(**DISTINCT** p.category, ', ') **AS** 購入カテゴリ一覧,  
MAX(s.order\_date) **AS** 最終購入日,  
SUM(s.quantity \* p.price) **AS** 累計購入金額

**FROM** 'data/customers.csv' **AS** c

**LEFT JOIN** 'data/sales.csv' **AS** s **ON** c.customer\_id = s.customer\_id

**LEFT JOIN** 'data/products.csv' **AS** p **ON** s.product\_id = p.product\_id

**GROUP BY** c.customer\_id, c.customer\_name

**ORDER BY** 累計購入金額 **DESC**;

# ● 本日のまとめ

---

今日学んだこと：

- ✓ コマンドの組み合わせで複雑な分析が可能
- ✓ LEFT JOINで「存在しないデータ」も発見
- ✓ 実務で使えるクエリのパターンを習得
- ✓ ビジネスの意思決定に役立つデータ抽出

# ● 8回の勉強会で学んだこと

---

- ✓ 基本的なデータ抽出 (SELECT、WHERE)
- ✓ データの並び替え (ORDER BY)
- ✓ 集計処理 (COUNT、SUM、AVG)
- ✓ グループ化 (GROUP BY)
- ✓ テーブル結合 (INNER JOIN、LEFT JOIN)
- ✓ 実践的な組み合わせ

# SQLの基本的な考え方

何を見たいか (SELECT)

どこから取るか (FROM)

どんな条件か (WHERE)

どう整理するか (GROUP BY、ORDER BY)

どう見せるか (AS、集計関数)

# ● 修了おめでとうございます！

---

これで基本的なSQLは書けるようになりました。

実務では：

エラーを恐れずに試す

結果を確認しながら調整

分からないことは調べる・聞く

**SQLは「データと対話する言語」です。**

たくさん書いて、慣れていきましょう！



# ● 今後の学習リソース

---

## 公式ドキュメント

DuckDB Documentation

## 練習サイト

SQLZoo

HackerRank SQL

LeetCode Database

# ● 追加演習問題

---

## 問題1：LEFT JOINの基本

以下のLEFT JOINを実行：

- 1. 全顧客の一覧（購入履歴の有無に関わらず）
- あなたの答えをここに書いてください
  
- 2. 全商品の売上状況（売れていない商品も含む）
- あなたの答えをここに書いてください
  
- 3. 顧客ごとの総購入金額（購入なしは0円として）
- あなたの答えをここに書いてください

## 問題2：データ品質チェック

LEFT JOINを使ったデータの整合性確認：

- 1. sales.csvに存在するが、customers.csvに存在しない顧客IDを探す
- ヒント：RIGHT JOINまたは別のアプローチ
  
- 2. 一度も売れていない商品のリスト（商品名、価格、カテゴリ付き）
- あなたの答えをここに書いてください

## 問題3：総合分析クエリ

以下のビジネス要件をSQLで実現：

- 1. 顧客セグメンテーション
- 全顧客を以下に分類：
  - - VIP顧客：購入金額10万円以上
  - - 通常顧客：購入あり、10万円未満
  - - 休眠顧客：購入なし
- 2. 商品推奨リスト
- 各顧客に対して「まだ買っていない商品」のリストを作成
- ヒント：CROSS JOINとLEFT JOINの組み合わせ

## 問題4：レポート作成

経営陣向けの総合レポート：

- 以下の情報を含む「顧客価値分析レポート」を作成：
- - 顧客名
- - 登録日
- - 総購入回数
- - 総購入金額
- - 最終購入日
- - お気に入りカテゴリ（最も多く購入したカテゴリ）
- - 顧客ランク（VIP/通常/休眠）

## チャレンジ問題：RFM分析

- RFM分析の簡易版を実装：
- R (Recency)： 最終購入日からの経過日数
- F (Frequency)： 購入頻度
- M (Monetary)： 購入金額
- 各指標でスコアリングし、優良顧客を特定



## 実践問題：ダッシュボード用クエリ集

- 1. KPIサマリー（1つのクエリで）
  - - アクティブ顧客数
  - - 総売上金額
  - - 平均購入単価
  - - 売れ筋商品TOP3
- 2. 期間比較分析
  - 前半（1/15-1/22）と後半（1/23-1/29）の売上比較
- 3. カテゴリ別パフォーマンス
  - 各カテゴリの売上、顧客数、平均単価

## デバッグ練習：複雑なクエリ

- 以下のクエリを修正して、意図通りに動作させてください：
- 意図：各顧客の購入商品一覧（商品名、合計数量、合計金額）

```
SELECT
    c.customer_name,
    p.product_name,
    SUM(s.quantity),
    SUM(s.quantity * p.price)
FROM 'data/customers.csv' c
LEFT JOIN 'data/sales.csv' s
LEFT JOIN 'data/products.csv' p
GROUP BY c.customer_name;
```





## 応用：ウィンドウ関数の代替

- ウィンドウ関数を使わずに以下を実現：
- 1. 各顧客の購入履歴に「何回目の購入か」を付ける
- 2. 日別売上の累計を計算
- 3. 商品売上ランキングに順位を付ける



## 総合演習：ECサイト分析

- 以下の分析を全て実行：
- 1. コホート分析：登録月別の顧客の購買行動
- 2. バスケット分析：よく一緒に買われる商品の組み合わせ
- 3. 離脱顧客分析：最後の購入から時間が経っている顧客
- 4. 商品ライフサイクル：商品ごとの売上推移

## 最終課題：SQLマスターへの道

- 以下の要件を満たす「月次売上レポート」を作成：
- 
- 【要件】
- 1. 全顧客の当月売上サマリー（購入なしも0円で表示）
- 2. 新規顧客（当月登録）の売上貢献度
- 3. カテゴリ別の売上構成比
- 4. 前月比較（仮想的に前月データがあると仮定）
- 5. 顧客別の成長率ランキング
- 
- これを可能な限り少ないクエリ数で実現してください

## 考察問題

- 以下について、SQLで分析し、ビジネス上の示唆を考えてください：
- 1. なぜ特定の顧客は購入していないのか？
- 2. どの商品の組み合わせが売上を最大化するか？
- 3. 顧客の離脱を防ぐために、どんなデータが必要か？

## さらなる学習のために

- 今回学ばなかった以下の構文を調べて、試してみましょう：
- 1. WITH RECURSIVE（階層データの処理）
- 2. PIVOT/UNPIVOT（データの転置）
- 3. WINDOW関数（ROW\_NUMBER, RANK, LAG/LEAD）
- 4. JSON関数（最近のDBMSでサポート）
- 5. 正規表現を使った検索（REGEXP）