📊 대시보드 데이터 구조 및 조인 가이드

> **중요**: 이 문서는 데이터 처리 시 자주 발생하는 실수를 방지하기 위한 필수 가이드입니다.

📁 데이터 디렉토리 구조

```
exceldashboard/
├── 주문 데이터/
│   ├── 20251219.csv
│   ├── 20251226.csv
│   └── 20260102.csv
├── 결제 데이터/
│   ├── 20251219.csv
│   ├── 20251226.csv
│   └── 20260102.csv
└── 누적 결제 데이터/
    ├── 20251219.csv
    ├── 20251226.csv
    └── 20260102.csv
```

1️⃣ 주문 데이터 (Order Data)

파일 위치

`주문 데이터/YYYYMMDD.csv`

컬럼 구조

```
pg_yn, shop_code, shop_name, pos_code,
sol_pay_promotion_yn, nice_pay_promotion_yn,
ins_datetime, formatted_date, company_name, prev_company_name,
shop_status, formatted_date,
device_count, table_count,
total_count_all, order_count_all,
total_count_no_pos, total_price_no_pos,
order_count_no_pos, price_no_pos
```

주요 컬럼 설명

컬럼명	타입	설명	비고
`pg_yn`	`'선불' | '후불'`	결제 유형	**필터링 필수**
`shop_code`	`string`	매장 코드	**조인 키**

컬럼명	타입	설명	비고
shop_name	string	매장명	
pos_code	string	POS 시스템 코드	
sol_pay_promotion_yn	'O' | 'X'	쏠페이 프로모션 참여 여부	
nice_pay_promotion_yn	'O' | 'X'	카카오페이 프로모션 참여 여부	
ins_datetime	string	매장 등록 일시	형식: YYYY-MM-DD HH:mm:ss
company_name	string	회사명	
prev_company_name	string	대리점명	- 또는 빈값이면 직영업
shop_status	'이용' | '이용대기' | '종료'	매장 상태	
device_count	number	디바이스 수	
table_count	number	테이블 수	
order_count_no_pos	number	**메뉴판앱 주문 건수** (주간)	POS 제외
price_no_pos	number	**메뉴판앱 주문 금액** (주간)	POS 제외
total_count_no_pos	number	메뉴판앱 총 주문 건수 (누적)	
total_price_no_pos	number	메뉴판앱 총 주문 금액 (누적)	

⚠ 중요 특성

1. **한 매장이 여러 행으로 존재할 수 있음** (예: POS 코드별로 분리)
2. 매장별로 데이터를 집계할 때는 **반드시 `shop_code`로 그룹화** 필요
3. `order_count_no_pos`와 `price_no_pos`는 **주문 데이터**이지 **결제 데이터**가 아님

---

2 결제 데이터 (Payment Data - 주간)

파일 위치

`결제 데이터/YYYYMMDD.csv`

컬럼 구조

```
pg_yn, shop_code, shop_name,
count, total_price,
sol_pay_amt, sol_pay_count,
kakao_money_amt, kakao_money_count
```

주요 컬럼 설명

컬럼명	타입	설명	비고
pg_yn	'선불' | '후불'	결제 유형	**있음** (주문 데이터와 조인 가능)
shop_code	string	매장 코드	**조인 키**
shop_name	string	매장명	
count	number	**주간 결제 건수**	
total_price	number	**주간 결제 금액**	
sol_pay_count	number	쏠페이 결제 건수	
sol_pay_amt	number	쏠페이 결제 금액	
kakao_money_count	number	카카오머니 결제 건수	
kakao_money_amt	number	카카오머니 결제 금액	

🔑 카카오페이 vs 카카오머니 구분

중요: 카카오페이와 카카오머니는 다릅니다!

카카오페이 결제

- **정의**: 나이스 프로모션(`nice_pay_promotion_yn === 'O'`) 매장의 결제
- **선불**: 카카오페이 선불 결제
- **후불**: 카카오 후불 결제
- **집계 방법**:
  1. 주문 데이터에서 `nice_pay_promotion_yn === 'O'` && `pg_yn === '선불'` 매장 추출
  2. 해당 매장들의 **전체 결제** (`count`, `total_price`) 사용

카카오머니 결제

- **정의**: 카카오페이 결제 중 **카카오머니**로 결제한 것
- **특징**: 카카오페이 내에서 충전된 머니로 결제
- **집계 방법**: `kakao_money_count`, `kakao_money_amt` 사용

관계도

```
카카오페이 결제 (count, total_price)
├── 카카오머니 결제 (kakao_money_count, kakao_money_amt)
```

```
└── 기타 결제수단 결제 (count - kakao_money_count)
```

예시:

- 매장 A의 전체 결제: 100건, 1,000,000원
- 그 중 카카오머니: 20건, 150,000원
- → 카카오페이 결제: 100건, 1,000,000원 ☑
- → 카카오머니 결제: 20건, 150,000원 ☑

⚠ 중요 특성

1. **매장당 1행** (주문 데이터와 달리 그룹화 불필요)
2. `pg_yn` 필드 존재: 선불/후불 구분 가능
3. **주간 데이터**: 해당 주의 결제만 포함

---

3 누적 결제 데이터 (Cumulative Payment Data)

파일 위치

`누적 결제 데이터/YYYYMMDD.csv`

컬럼 구조

```
pg_yn, shop_code, shop_name,
count, total_price,
sol_pay_amt, sol_pay_count,
kakao_money_amt, kakao_money_count
```

주요 컬럼 설명

결제 데이터와 동일한 구조이지만, **누적 값**

컬럼명	설명
`count`	**누적 결제 건수** (전체 기간)
`total_price`	**누적 결제 금액** (전체 기간)

⚠ 중요 특성

1. 결제 데이터와 동일한 구조, 값만 누적
2. **주간 vs 누적** 구분 필요

🔗 데이터 조인 가이드

조인 키

모든 데이터는 **`shop_code`**로 조인

조인이 필요한 이유

1. 프로모션 정보는 주문 데이터에만 존재

```
주문 데이터:
- sol_pay_promotion_yn (쏠페이 프로모션 여부)
- nice_pay_promotion_yn (카카오페이 프로모션 여부)

결제 데이터:
- 프로모션 정보 없음
```

결론: 프로모션 매장의 결제 데이터를 가져오려면 **반드시 조인 필요**

2. 결제 데이터는 결제 데이터에만 존재

```
주문 데이터:
- order_count_no_pos (주문 건수)
- price_no_pos (주문 금액)
- ❌ 실제 결제 건수/금액 없음

결제 데이터:
- count (결제 건수)
- total_price (결제 금액)
```

결론: 결제 통계를 내려면 **결제 데이터를 사용**해야 함

⚠️ 자주 하는 실수와 올바른 방법

❌ 실수 1: 주문 데이터로 결제 통계 계산

```
// ❌ 잘못된 방법
const paymentCount = orders.reduce((sum, o) => sum + o.order_count_no_pos, 0);
const paymentAmount = orders.reduce((sum, o) => sum + o.price_no_pos, 0);
```

```
// ☑ 올바른 방법
const paymentCount = payments.reduce((sum, p) => sum + p.count, 0);
const paymentAmount = payments.reduce((sum, p) => sum + p.total_price, 0);
```

이유: `order_count_no_pos`는 **주문 건수**이지 **결제 건수**가 아님

❌ 실수 2: 선불/후불 구분 없이 프로모션 매장 필터링

```
// ❌ 잘못된 방법 - 후불도 포함됨
const kakaoPayShopCodes = new Set(
  orders.filter(o => o.nice_pay_promotion_yn === 'O').map(o => o.shop_code)
);
const payments = paymentData.filter(p => kakaoPayShopCodes.has(p.shop_code));
```

```
// ☑ 올바른 방법 - 선불만 필터링
const kakaoPayPrepaidShopCodes = new Set(
  orders
    .filter(o => o.nice_pay_promotion_yn === 'O' && o.pg_yn === '선불')
    .map(o => o.shop_code)
);
const payments = paymentData.filter(p =>
kakaoPayPrepaidShopCodes.has(p.shop_code));
```

이유: 카카오페이 선불 활성화 현황에서는 **선불 매장만** 집계해야 함

❌ 실수 3: 주문 데이터 그룹화 누락

```
// ❌ 잘못된 방법 - 중복 카운트 발생
const totalOrders = orders
  .filter(o => o.pg_yn === '선불')
  .reduce((sum, o) => sum + o.order_count_no_pos, 0);
```

```
// ☑ 올바른 방법 - 매장별로 그룹화 후 합산
const shopMap = new Map<string, { orderCount: number }>();
orders.filter(o => o.pg_yn === '선불').forEach(o => {
  const existing = shopMap.get(o.shop_code) || { orderCount: 0 };
  existing.orderCount += o.order_count_no_pos;
  shopMap.set(o.shop_code, existing);
});
const totalOrders = Array.from(shopMap.values())
  .reduce((sum, s) => sum + s.orderCount, 0);
```

이유: 주문 데이터는 한 매장이 여러 행으로 존재할 수 있음

📋 실전 예제

예제 1: 카카오페이 선불 활성화 현황 계산

```
// Step 1: 주문 데이터에서 카카오페이 선불 이용 매장 코드 추출
const kakaoPayPrepaidShopCodes = new Set(
  currentOrders
    .filter(o =>
      o.nice_pay_promotion_yn === 'O' &&  // 카카오페이 프로모션
      o.pg_yn === '선불' &&                // 선불
      o.shop_status === '이용'             // 이용 중
    )
    .map(o => o.shop_code)
);

// Step 2: 결제 데이터에서 해당 매장들의 결제 정보만 필터링
const kakaoPayPayments = currentPayments.filter(p =>
  kakaoPayPrepaidShopCodes.has(p.shop_code)
);

// Step 3: 통계 계산
const stats = {
  activatedShops: kakaoPayPayments.filter(p => p.count > 0).length,
  paymentCount: kakaoPayPayments.reduce((sum, p) => sum + p.count, 0),
  paymentAmount: kakaoPayPayments
    .filter(p => p.count > 0)
    .reduce((sum, p) => sum + p.total_price, 0),
};
```

예제 2: 대리점별 실적 계산

```
// Step 1: 주문 데이터를 대리점별로 그룹화
const agencyMap = new Map<string, OrderData[]>();
currentOrders.forEach(order => {
  const agencyName = order.prev_company_name || '직영업';
  if (!agencyMap.has(agencyName)) {
    agencyMap.set(agencyName, []);
  }
  agencyMap.get(agencyName)!.push(order);
});

// Step 2: 각 대리점별 통계 계산
for (const [agencyName, shops] of agencyMap) {
  // 매장별로 그룹화 (중복 제거)
  const uniqueShops = new Map<string, OrderData>();
  shops.forEach(shop => {
    if (!uniqueShops.has(shop.shop_code)) {
      uniqueShops.set(shop.shop_code, shop);
    }
  });

  const totalShops = uniqueShops.size;
  const activeShops = Array.from(uniqueShops.values())
```

```
    .filter(s => s.shop_status === '이용').length;
}
```

🎯 핵심 체크리스트

- ☐ 결제 통계를 계산할 때 **결제 데이터** 사용 (주문 데이터 ❌)
- ☐ 프로모션 매장을 필터링할 때 **선불/후불 구분** 필수
- ☐ 주문 데이터 집계 시 **매장별 그룹화** 필수
- ☐ 조인 시 `shop_code` 사용
- ☐ 주간 vs 누적 데이터 **구분** 확인
- ☐ `pg_yn` 필드로 **선불/후불 필터링**
- ☐ 프로모션 정보는 **주문 데이터**에서만 가져오기

📌 TypeScript 타입 정의

```typescript
// 주문 데이터
interface OrderData {
  pg_yn: '선불' | '후불';
  shop_code: string;
  shop_name: string;
  pos_code: string;
  sol_pay_promotion_yn: 'O' | 'X';
  nice_pay_promotion_yn: 'O' | 'X';
  ins_datetime: string;
  company_name: string;
  prev_company_name: string;
  shop_status: '이용' | '이용대기' | '종료';
  device_count: number;
  table_count: number;
  order_count_no_pos: number;  // 주문 건수 (주간)
  price_no_pos: number;        // 주문 금액 (주간)
  total_count_no_pos: number;  // 주문 건수 (누적)
  total_price_no_pos: number;  // 주문 금액 (누적)
}

// 결제 데이터 (주간 & 누적)
interface PaymentData {
  pg_yn: '선불' | '후불';
  shop_code: string;
  shop_name: string;
  count: number;              // 결제 건수
  total_price: number;        // 결제 금액
  sol_pay_count: number;      // 쏠페이 건수
  sol_pay_amt: number;        // 쏠페이 금액
  kakao_money_count: number;  // 카카오머니 건수
  kakao_money_amt: number;    // 카카오머니 금액
}
```

최종 업데이트: 2026-01-08
문서 버전: 1.0