DB

1. Business Rules Summary

- 프로젝트 목표 : 편의점 체인 본사 운영 및 운영 체계를 반영한 ERD 설계
- ERD 요약 :
  - 모델에서는 직영점과 프랜차이즈 지점을 구분하여 관리할 수 있으며, 매장별로 자체적인 운영 시간과 재고 관리가 가능합니다. 일부 점포는 24시 영업이며, 매장 형태나 위치에 따라 특정 제품에 대한 판매 여부가 달라질 수 있습니다.
  - 고객 측면에서는 모든 구매자가 고객 정보에 등록되지는 않습니다. 회원과 비회원을 구분하여 관리하여 추후에 회원/비회원 전환율에 사용할 수 있습니다. 회원 고객은 멤버십을 통하여 등급이 부여됩니다. (예: GOLD ,SILVER) 등급에 따른 할인 혜택과 분석이 가능할 것입니다.
  - 판매 이력은 단순 거래 기록이상으로 구매 패턴,판매 수량,매출 비교, 마케팅 등에 활용할 수 있도록 설계하였습니다. 점포에서 재고가 일정 수준으로 떨어질 시 자동 재주문이 되도록 재고 임계값을 세팅하였습니다.

2. Entity and Relationship Justification

Entity

- Entity 별 속성 Table

Entity	Primary Key	Attributes	Foreign Key
Store	store_id	name, address, open_time, close_time, ownership_type	-
Product	upc	name, brand, package_type, size, price, category	vendor_id
Vendor	vendor_id	name, phone_number, email	-
Inventory	(store_id, upc)	quantity, reorder_threshold, reorder_amount	store_id, upc
Customer	customer_id	name, phone, email	-
Membership	membership_id	level, join_date, discount_rate, customer_id	customer_id
SaleTransaction	saleTransaction_id	transaction_time, payment_method	store_id, customer_id (optional)
SaleDetail	(saleTransaction_id, upc)	quantity, unit_price, discount_rate	saleTransaction_id, upc

- Entity별 세부 설명

- **Store**

  편의점 체인 시스템의 물리적 운영 단위로, 모든 비즈니스 활동이 실제로 이루어지는 공간을 의미합니다.각 매장은 고유한 `store_id`로 식별됩니다. 이름, 주소, 운영 시간(open_time, close_time) 등의 속성을 통해 위치 및 영업 정보를 관리할 수 있습니다.

  또한 Store는 Inventory,SaleTransaction 등 핵심 트랜잭션과의 연결점이 되어,매장별 재고 관리,판매 이력 집계,고객 유입 분석 등에 활용됩니다.

  - 주요 Attribute 설명

  `ownership_type` : 속성을 통해 프랜차이즈와 직영 여부를 구분하기 위한 필드로, 매장 운영 형태에 따른 정책 차별화나 성과 비교를 할 수 있는 근거 데이터로 활용될 수 있습니다. 매장은 제품과의 관계에서 재고를 관리하며, 판매 기록의 출발점 역할을 합니다.

  `open_time` & `close_time` : 24시 운영 매장은 open_time과 close_time을 00:00으로 설정하여 구분할 수 있습니다.

- **Product**

  편의점에서 판매되는 모든 개별 품목을 추상화한 엔티티입니다. upc,제품명, 브랜드, 패키지 타입, 용량(사이즈), 가격 등 다양한 속성을 포함합니다.
  각 제품은 여러 매장에서 판매될 수 있으며, 동일한 제품이라 하더라도 매장에 따라 가격, 진열 여부, 재고 수량이 달라질 수 있습니다.오직 한 공급업체에 의해 공급되도록 가정하였습니다.

  제품 정보는 재고 파악, 매출 집계, 추천 알고리즘 등 다양한 기능의 기반이 됩니다.

  - 주요 attribute 설명

  `upc` :(Universal Product Code), 각 제품의 고유 식별자

  `category` : 식품, 음료, 비식품 등으로 구분하기 위한 속성으로, 카테고리별 매출, 재고 회전율 등에 필요하다고 판단하여 추가하였습니다.

  `vendor_id` : 외래키로 보유하여, 해당 제품을 공급하는 업체와 직접적으로 연결되며, 공급업체별 납품 제품 수, 품목별 공급 안정성 등의 분석이 가능합니다.

  Product 엔티티는 `Inventory`, `SaleDetail`, `Vendor` 엔티티와 밀접하게 연결되어 있어,**제품 중심적인 분석**이 가능하도록 핵심 역할을 수행합니다.

- **Vendor**

  `vendor_id`를 기본 키로 가지며, 업체명, 연락처 등의 속성을 포함합니다. 제품 공급업체로, 하나의 Vendor는 여러 개의 제품을 공급할 수 있는 **1:N 관계** 구조입니다.

- **Inventory**

  Store와 Product 사이의 다대다 관계를 분해하기 위해 도입된
  중간 엔티티입니다. 하나의 매장에 어떤 제품이 몇 개 남아 있는지를 추적하며, `quantity`, `reorder_threshold`, `reorder_amount` 등의 속성을 통해 **재고 부족 알림**, **자동 발주 시스템**을 구현할 수 있습니다. 이 엔티티는 제품별 재고 수량, 입고 상태, 재고 회전율 등을 분석하는 데 필요한 데이터를 제공합니다

  - 주요 attribute 설명

  `reorder_threshold` & `reorder_amount`: quantity가 해당 reorder_threshold 낮을 경우 reorder_amount 만큼 자동 재 주문을 하도록 설정할 수 있습니다.

- **Customer**

  멤버십 및 구매 분석을 위해 설계된 엔티티입니다. 고객 ID로 식별하며, 추후 마케팅에 활용하기 위한 이름, 연락처 등을 개인 정보를 포함하였습니다. 이번 프로젝트에서는

  편의점이라는 특성상, **비회원 고객**이 전체 구매자의 다수를 차지할 수도 있으므로, 고객 정보는 **선택적으로 등록**될 수 있게 설계되었습니다.

  회원 고객일 경우 Membership과의 1:1 관계를 통해 **등급 기반 마케팅**, **로열티 프로그램 설계**, **고객 맞춤형 추천** 등으로 확장됩니다.

  거래는 고객 없이도 발생할 수 있도록 되어 있어, **비회원 트랜잭션**도 시스템에 기록 가능합니다.

  이를 통해 본사는 회원/비회원의 구매 행동 차이, 할인 효과 차이 등을 정밀하게 분석할 수 있습니다.

- **Membership**

  고객의 멤버십 상태 및 등급(GOLD, SILVER 등)을 관리하는 엔티티입니다.
  각 Membership은 정확히 한 명의 고객에 대응하며, `level`, `join_date`, `discount_rate` 등의 속성을 가집니다. 이를 통해 고객별 혜택을 다르게 적용하거나, 등급 별 고객의 구매 패턴을 분석해 **상품 추천, 혜택 설계, 로열티 프로그램** 등에 활용할 수 있습니다.
  Customer 과는 **1:1 identifying relationship**으로 연결됩니다.

  - 주요 attribute 설명

  `level` : 등급을 나타내며 예를 들어 GOLD, SILVER 처럼 구분됩니다.

  `join_date` :멤버십 가입일입니다.

  `discount_rate`: 해당 등급에 부여된 할인 비율입니다.

- **SaleTransaction**

  SaleTransaction은 편의점에서 실제 결제가 발생한 **하나의 거래 단위**를 나타내는 엔티티입니다.

  `saleTransaction_id`로 식별되며, 거래 발생 시점의 날짜 및 시간, 결제 수단 등의 정보를 포함합니다.

  거래는 반드시 store_id와 연결되어 어떤 매장에서 발생했는지를 명확히 하며,옵션으로 customer_id가 포함되어 고객 기반 분석도 가능하게 합니다.

  본 프로젝트에서는 정규화 원칙을 준수하기 위하여 할인 총액 , 최종 결제 금액 등 **계산으로 유도 가능한 파생 속성은 SaleTransaction 엔티티에 포함하지 않았습니다.**

  각 거래의 총 금액 및 할인 총액은 연결된 `SaleDetail`엔티티의 unit_price, quantity, discount_rate 속성을 기반으로 다음과 같이 계산할 수 있습니다

  > 할인 총액 = ∑ (unit_price × quantity × discount_rate)
  >
  > 최종 결제 금액 = ∑ (unit_price × quantity) - 할인 총액

  SaleTransaction은 특정 매장에서 발생하며(store_id), 선택적으로 특정 고객(customer_id)과 연결될 수 있습니다.또한 하나의 거래는 여러 개의 상품 항목을 포함하므로, **SaleDetail과 1:N 관계**로 연결됩니다.

  시간대별 매출 분석, 매장별 판매 실적 비교, 고객 구매 패턴 파악 등 다양한 비즈니스 분석에 핵심 데이터로 활용됩니다.

- **SaleDetail**

  SaleDetail 엔티티는 SaleTransaction과 Product 간의 **구체적인 구매 항목**을 나타냅니다.

  즉, 하나의 거래에 어떤 제품이 몇 개나 포함되었는지를 표현하는 역할로, 복합키 `(saleTransaction_id, upc)`를 가집니다.

  속성으로는 `quantity`, `unit_price`, `discount_rate`가 있으며,

  각 항목별 실제 판매 수량, 단가, 할인율을 개별적으로 관리할 수 있도록 구성되어 있습니다.

Relationship

- Relationship 요약 Table

관계 이름	연결된 엔티티	Cardinality	Participation	Weak Entity
Inventory_Store	Inventory – Store	N:1	Total (Inventory 측)	❌
Product_Inventory	Inventory – Product	N:1	Total (Inventory 측)	❌
Supplies	Vendor – Product	1:N	Partial (Product 측)	❌
Customer_Membership	Customer – Membership	1:1	Total (Membership 측), Partial (Customer 측)	☑️(Membership)
Customer_SaleTransaction	Customer – SaleTransaction	1:N (optional)	Partial (SaleTransaction 측)	❌
SaleTransaction_Store	SaleTransaction – Store	N:1	Total (SaleTransaction 측)	❌
ProcessesTransaction	SaleTransaction – SaleDetail	1:N	Total (SaleDetail 측)	☑️ (SaleDetail)
RefersToProduct	SaleDetail – Product	N:1	Total (SaleDetail 측)	❌

- Relation별 세부 설명
  - Inventory_Store
    - 각 재고 항목(Inventory)은 반드시 하나의 매장(Store)에 소속되며, 이는 **Total participation 관계**로 표현됩니다. 즉, 모든 재고는 매장과 연결되어 있어야 하며, 매장 없이 존재할 수 없습니다. 반면, 한 매장은 여러 재고 항목을 가질 수 있으므로 cardinality는 **N:1**입니다.이를 통해 같은 제품이라도 매장마다 재고 수량, 재주문 임계값 등이 다르게 관리될 수 있으며,매장 단위 재고 분석 및 발주 전략에 핵심적인 정보를 제공합니다.
  - Product_Inventory
    - 하나의 재고 항목은 반드시 하나의 제품(Product)에 대응되므로, Inventory는 Product에 대해 **N:1 관계**, 그리고**Total Participation**을 갖습니다. 즉, 모든 Inventory는 반드시 하나의 Product에 연결되어야 하며, 제품 없이 독립적으로 존재할 수 없습니다.이 구조는 동일한 제품이 여러 매장에서 독립적으로 판매 및 관리될 수 있음을 나타내며,제품별 재고 현황 집계, 평균 재고량 계산, 회전율 분석에 필요한 정보를 제공합니다.
  - Supplies
    - 하나의 공급업체(Vendor)는 여러 제품(Product)을 납품할 수 있고, 하나의 제품은 단 하나의 공급업체에만 연결된다고 가정하여 **1:N 관계**로 모델링합니다. Product 측은 **Partial Participation**으로, 일부 제품만 특정 Vendor에 연결될 수도 있음을 의미합니다.이 구조는 공급처별 제품군 관리, 공급업체 성과 비교, 리스크 분석 등을 가능하게 해줍니다.
  - Customer_Membership
    - 고객(Customer)은 최대 1개의 멤버십(Membership)을 가질 수 있고, 모든 고객이 멤버십을 갖는 것은 아니므로 Customer 측은 **Partial Participation**입니다. 하지만 Membership은 반드시 Customer에 의존하며, 단독으로 존재할 수 없기 때문에 **Weak Entity**이며 **Total Participation**입니다. 이 관계를 통해 회원/비회원 비교 분석, 등급별 혜택 관리, 마케팅 타겟팅에 필요한 데이터를 생성합니다.
  - Customer_SaleTransaction
    - 고객은 여러 판매 거래(SaleTransaction)를 할 수 있으며, 거래는 고객 없이도 발생할 수 있는 구조이기 때문에 SaleTransaction 측은 **Partial Participation** 입니다. 이는 비회원 구매 나 익명 거래를 반영합니다.이 구조는 고객 기반 구매 이력 분석, 재구매율 측정, 비회원 거래 통계 분석 등에서 중요한 역할을 합니다.
  - SaleTransaction_Store
    - 모든 판매 거래는 반드시 특정 매장에서 발생해야 합니다. 따라서 SaleTransaction 측은 **Total Participation**입니다. 각 매장은 여러 판매 거래를 가질 수 있으므로 이 관계의 cardinality는 **N:1**입니다. 이를 통해 매장별 일/주/월간 매출, 시간대별 트랜잭션 수, 지점 간 성과 비교 등을 정확히 측정할 수 있습니다.
  - ProcessesTransaction
    - 하나의 판매 거래(SaleTransaction)는 여러 제품 항목(SaleDetail)을 포함할 수 있습니다. 각 SaleDetail은 반드시 상위 거래와 연결되어야 하므로, **Total Participation**이며, 독립적으로 존재할 수 없기 때문에 **Weak Entity**로 간주합니다. 예컨대 하나의 결제에서 음료, 도시락, 디저트를 함께 구매한 경우를 모두 기록하며, 이 관계를 통해 장바구니 분석, 동시 구매 패턴, 1회 거래당 평균 구매 품목 수 등을 추출할 수 있습니다.
  - RefersToProduct
    - 각 거래 항목(SaleDetail)은 특정 제품(Product)을 참조하기 때문에 SaleDetail 측은 **Total Participation**입니다. 하나의 제품은 여러 SaleDetail에 반복적으로 등장할 수 있으므로 cardinality는 **N:1**입니다. 이 관계는 제품별 판매 수량 집계, 제품 단가 변동 추적, 상품군별 판매율 분석 등에 핵심적인 역할을 합니다.

4. Query Support Description

Q.

Product Availability: "Which stores currently carry a certain product (by UPC, name, or brand), and how much inventory do they have?"

A.

Inventory Entity에서 `store_id`와 `upc`를 조건으로 검색한 후 quantity attribute로 현재 수량을 확인할 수 있습니다.

Ex.

Query: "강남1호점에서 '코카콜라 500ml'의 재고는 몇 개인가요?"
→
`store_id = 'S001'`, `upc = '8801234567890'` 조건으로 검색 → `quantity = 45`

Q.

Top-Selling Items: "Which products have the highest sales volume in each store over the past month?"

A.

`SaleTransaction`에서 `transaction_time` 기준으로 최근 한 달간의 거래를 필터링한 뒤,해당 거래들과 연결된 `SaleDetail`에서 `upc`별로 `quantity`를 집계하여 가장 많이 팔린 제품을 도출함.

Ex.

Query: "3월에 가장 많이 팔린 제품은?"

→ `SaleTransaction.transaction_time BETWEEN '2025-03-01' AND '2025-03-31'`

→ 해당 거래들의 `SaleDetail`을 조회 후, `upc`별 `SUM(quantity)`

→ 결과: `'8801111222333'` (삼각김밥 참치마요) → `총 312개 판매`

Q.

Store Performance: "Which store has generated the highest overall revenue this quarter?"

A.

1. `SaleTransaction`에서 `store_id` 기준으로 를 합산
2. `GROUP BY store_id` 후 `SUM()` 기준으로 내림차순 정렬하여 상위 매장 확인

Ex.

Query: "이번 분기 동안 매출 1위 매장은?"

→ `SaleTransaction.transaction_time BETWEEN '2025-01-01' AND '2025-03-31'`

→ `GROUP BY store_id` → `SUM()`

→ 결과: `store_id = 'S005'` (홍대2호점), `총매출: 24,320,000원`

Q.

Vendor Statistics: "Which vendor supplies the most products across the chain, and how many total units have been sold?"

A. .

`Supplies` 관계 테이블에서 `vendor_id`별로 공급한 `upc` 수를 `COUNT(DISTINCT upc)`로 집계

Ex.

Query: "가장 많은 제품을 취급하는 공급업체는?"

- →
```
SELECT vendor_id, COUNT(*) AS product_count
FROM Product
GROUP BY vendor_id
ORDER BY product_count DESC;
```

→ 결과: `vendor_id = 'V102'` , `공급 제품 수: 54개`

Q.

Inventory Reorder Alerts: "Which products in each store are below the reorder threshold and need restocking?"

A.

`Inventory`에서 `quantity < reorder_threshold` 조건을 만족하는 레코드를 필터링

점포별, 제품별로 어떤 항목이 재고 부족 상태인지 확인 가능

Ex.

Query: "재주문이 필요한 제품은?"
→ 조건:
`quantity < reorder_threshold`

→ 결과: `quantity: 7` , `reorder_threshold: 10`

Q.

Customer Purchase Patterns: "List the top 3 items that loyalty program customers typically purchase with coffee."

A.

1. `Membership`에서 `level = 'GOLD'`인 고객 조회
2. 해당 고객들의 `SaleTransaction` 추출
3. 거래 내 `SaleDetail`에서 **커피 포함된 거래** 필터
4. 해당 거래에서 **커피 외의 다른 제품들**을 추출하여 `upc`별 `SUM(quantity)`로 집계
5. `TOP 3` 결과 추출

Ex.

Query: "VIP가 커피와 함께 자주 사는 제품은?"

- → 고객 `C1001, C1005, C1012` (GOLD 고객)
- → 그들의 거래 중 커피(예: `upc = 20211999`)가 포함된 거래 ID 필터
- → 같은 거래 ID에서 다른 `upc` 제품 집계

  → 결과:

  1. `'20211994'` (크림빵) – 58회
  2. `'20211992'` (초콜릿바) – 41회
  3. `'20211995'` (샌드위치) – 36회

Q.

Franchise vs. Corporate Comparison: "Among franchise-owned stores, which one offers the widest variety of products, and how does that compare to corporate-owned stores?

A.

1. Store 엔티티에서 ownership_type이 "franchise" 또는 "corporate"인 매장 분리
2. Inventory를 통해 각 매장이 보유한 upc를 추출 (재고 수량과 무관하게 등록된 제품 기준)
3. GROUP BY store_id로 묶고, COUNT(DISTINCT upc)로 각 매장이 보유한 **제품 종류 수** 계산
4. 각각의 그룹에서 가장 다양한 제품을 보유한 매장을 추출하여 비교

Ex.

Query:"프랜차이즈 매장과 직영 매장 중 각각 제품 종류가 가장 많은 매장은 어디인가요?"

→ 결과 :

- 프랜차이즈 매장:
  - S103 → 제품 종류 수: 185
  - S081 → 제품 종류 수: 173
- 직영 매장:
  - S021 → 제품 종류 수: 161
  - S014 → 제품 종류 수: 150