BUILD WEEK 2

Siete stati assunti! Avete un senior di riferimento che vi guida! Ma si è ammalato. Avete una settimana, senza di lui, per creare un DB che permetta alla VendiCose SpA, famosissima società che gestisce supermercati, di gestire il flusso degli ordini per i magazzini e i punti vendita ad essi associati.

Cosa sapete?

Ogni categoria di prodotto (ad esempio, "alimentari", o "cosmetica") ha un livello di restock associato ad ogni magazzino. Questo significa che, quando le unità di un prodotto appartenente ad una determinata categoria scendono sotto una determinata soglia, in un determinato magazzino, bisogna effettuare un nuovo ordine.

Per tenere traccia in modo dinamico della quantità di prodotto presente in ogni magazzino, ogni qualvolta viene effettuata una vendita in un determinato negozio vengono aggiornati i dati. Sapete anche che ogni magazzino può servire più di un negozio.

Cosa dovete fare?

- Progettazione e successivo schema ER della base dati. Ricordatevi di indicare le chiavi e le relazioni

- Creazione della base dati tramite DDL (Suggerimento: potete chiedere ad una GenAI di generare per voi i dati da inserire)

-Scrittura delle query necessarie al funzionamento del sistema, in particolare:

1) Ogni qual volta un prodotto viene venduto in un negozio, qual è la query da eseguire per aggiornare le tabelle di riferimento?

2) Quali sono le query da eseguire per verificare quante unità di un prodotto ci sono un dato magazzino?

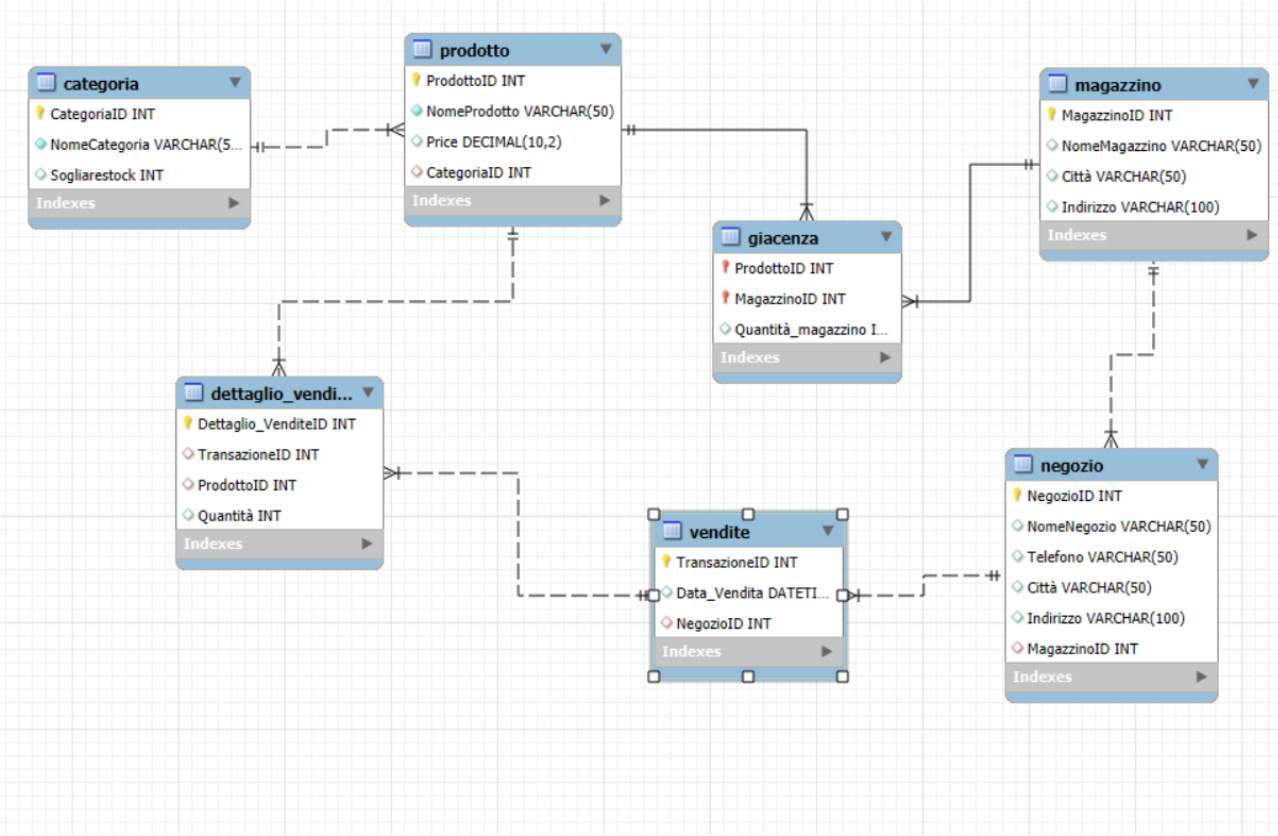
3) e per monitorare le soglie di restock?

TIPS

- Il flusso base delle informazioni parte dalle singole transazioni, che avvengono in un negozio il quale è associato ad un determinato magazzino

- Costruite la base dati seguendo il principio della normalizzazione (e della minimizzazione della ridondanza dei dati): avrete bisogno di più tabelle oltre alle tre che vengono suggerite dal flusso qui sopra.

SVOLGIMENTO



Per il popolamento del Database abbiamo scelto di inserire prima i dati riguardanti tutte le entità inerenti *"all'inventario"* dell'azienda e le strutture (quindi **prodotto, magazzino, categoria, negozio e giacenza**) e successivamente le singole vendite con i rispettivi dettagli, così da poter, dopo ogni acquisto, **aggiornare le quantità in giacenza** ed eventualmente avere la possibilità di controllare le soglie di restock:

```
INSERT INTO Categoria (NomeCategoria, Sogliarestock) VALUES

('Action Figures', 50), ('Puzzle', 30), ('Giochi Educativi', 40), ('Bambole', 60), ('Costruzioni', 70);


INSERT INTO Prodotto (NomeProdotto, Price, CategoriaID) VALUES

('Super Robot X', 29.99, 1), ('Puzzle Mondo 1000 Pezzi', 14.50, 2), ('Abaco Colorato', 9.99, 3), ('Bambola
Anna con Accessori', 24.90, 4), ('Blocchi Costruzioni Maxi', 39.99, 5), ('Puzzle Animali della Fattoria',
12.00, 2),

('Kit Esperimenti Base', 18.75, 3), ('Action Hero Blaze', 22.50, 1), ('Mattoncini Creativi 500 pezzi', 44.99,
5);


INSERT INTO Magazzino (NomeMagazzino, Città, Indirizzo) VALUES

('Magazzino Nord', 'Milano', 'Via Lombardia 45'),
```

```
('Magazzino Sud', 'Napoli', 'Via Vesuvio 12'),
('Magazzino Centro', 'Firenze', 'Via del Mercato 101');

INSERT INTO Negozio (NomeNegozio, Telefono, Città, Indirizzo, MagazzinoID) VALUES
('GiocaGiò Milano', '02-1234567', 'Milano', 'Corso Buenos Aires 33', 1),
('Bimbi Felici Napoli', '081-9876543', 'Napoli', 'Via Partenope 10', 2),
('Giochi&Cose Firenze', '055-334455', 'Firenze', 'Piazza della Libertà 88', 3);

INSERT INTO Giacenza (ProdottoID, MagazzinoID, Quantità_magazzino) VALUES
(1, 1, 200), (2, 1, 80), (3, 2, 120), (4, 2, 40), (5, 3, 160), (6, 1, 60), (7, 3, 100), (8, 2, 180), (9, 3,
240);
```

Questi i dati afferenti alle vendite e i dettagli vendita:

```
INSERT INTO Vendite (Data_Vendita, NegozioID) VALUES
('2025-06-15 10:30:00', 1), ('2025-06-15 15:45:00', 2), ('2025-06-16 11:10:00', 3),
('2025-06-16 17:20:00', 1);

INSERT INTO Dettaglio_Vendite (TransazioneID, ProdottoID, Quantità) VALUES
(1, 1, 4), (1, 2, 2),
(2, 4, 6), (2, 3, 2),
(3, 5, 4), (3, 7, 2),
(4, 8, 2), (4, 6, 4);
```

1) *Ogni qual volta un prodotto viene venduto in un negozio, qual è la query da eseguire per aggiornare le tabelle di riferimento?*

```
CREATE VIEW ID AS (SELECT
        dv.ProdottoID,
        n.MagazzinoID,
        dv.Quantità AS QuantitàVenduta
    FROM Dettaglio_Vendite dv
    JOIN Vendite v ON dv.TransazioneID = v.TransazioneID
    JOIN Negozio n ON v.NegozioID = n.NegozioID
    WHERE dv.TransazioneID = (select transazioneid from vendite order by transazioneid desc limit 1));
```

La **view** ID mostra quali **prodotti** e in quale **quantità** (in ogni vendita sono stati venduti 2 prodotti in quantita ***n***) sono stati **venduti nell'ultima transazione** registrata, indicando anche il **magazzino** associato al **negozio** dove è avvenuta la vendita.

Inserendo la **view** in una join con la tabella Giacenza otteniamo query che serve ad **aggiornare le quantità in magazzino** ogni volta che viene registrata una nuova vendita.

```
UPDATE GIACENZA G

JOIN (SELECT * FROM ID) AS SUB

ON g.ProdottoID = sub.ProdottoID AND g.MagazzinoID = sub.MagazzinoID

SET g.Quantità_magazzino = g.Quantità_magazzino - sub.QuantitàVenduta;
```

Piccola accortezza: Se nella transazione inseriamo anche le due INSERT, avremo l'impossibilità di poter aggiornare con i dati con cui appena aggiornato in precedenza a causa delle logiche dell'auto_increment impostato sulla chiave primaria delle vendite. Per ovviare a questo problema, inseriamo `(select transazioneid from vendite order by transazioneid desc limit 1),` al posto dei dati della transazioneid all'interno dell'INSERT del dettaglio_vendite, poiché così recuperiamo la transazioneid creata dall'autoincrement.

Inserendo questa query come update dopo ogni inserimento vendita e i suoi dettagli, riusciamo a tenere traccia costantemente di quelli che sono le quantità di prodotto all'interno di un dato magazzino.

2)Quali sono le query da eseguire per verificare quante unità di un prodotto ci sono un dato magazzino?

```
CREATE VIEW Anagrafica_prodotto AS (SELECT g.prodottoid, p.nomeprodotto, g.quantità_magazzino AS
Giacenza_prodotto, g.magazzinoid, m.nomemagazzino

FROM giacenza g

JOIN prodotto p ON g.prodottoid = p.prodottoid

JOIN magazzino m ON g.magazzinoid = m.magazzinoid ORDER BY prodottoid ASC);
```

Questa view restituisce una **panoramica aggiornata della quantità dei prodotti nei magazzini**, unendo informazioni da più tabelle.

Ad esempio, vogliamo vedere nel magazzino con id 2 la giacenza del prodotto con prodottoid 8:

```
SELECT * FROM anagrafica_prodotto

WHERE magazzinoid = 2 AND prodottoid = 8;
```

3) e per monitorare le soglie di restock?

```
CREATE VIEW Restock AS

              (SELECT g.prodottoid, p.nomeprodotto, g.magazzinoid, m.nomemagazzino, g.quantità_magazzino,
c.sogliarestock, IF ((g.quantità_magazzino - c.sogliarestock) > 0, 'OK', 'RESTOCK') AS Alert_Restock

FROM giacenza g

JOIN prodotto p ON g.prodottoid = p.prodottoid

JOIN magazzino m ON g.magazzinoid = m.magazzinoid
```

```
JOIN categoria c ON p.categoriaid = c.categoriaid

ORDER BY Alert_Restock DESC);


SELECT * FROM restock;
```

Questa **view** ci permette di visionare rapidamente l'**anagrafica del prodotto** e le sue **quantità** all'interno di **ogni magazzino**, mostrando quali sono i magazzini che **necessitano di un restock** di un dato prodotto **se la quantità in giacenza di questi ultimi è inferiore** a quella della **soglia prestabilita per il restock**. Qualora `g.quantità_magazzino - c.sogliarestock) > 0,` lo stato di giacenza è "OK", altrimenti l'Alert_Restock indica i prodotti per cui si necessita un "RESTOCK".