1) **Cosa si intende per database?**

Un database è una raccolta strutturata di dati, organizzati in maniera tale da facilitarne le operazioni di gestione e consultazione.

2) **Cos'è un DBMS?**

Un DBMS (Acronimo di Database Management System) è un software che consente di creare, gestire e manipolare database, facilitando l'interazione con il DB.

3) **Indica le principali clausole di uno statement SELECT in ordine di esecuzione logica. Descrivi per ciascuna delle clausole indicate la logica di funzionamento.**

FROM: Specifica le tabelle da cui verranno prelevati i dati interessati.

WHERE: Filtra i record in base a determinate condizioni.

GROUP BY: Raggruppa i dati in base a uno/più campi.

HAVING: Filtra i gruppi risultanti in base a condizioni specifiche.

SELECT: Seleziona i campi interessati da visualizzare come output.

ORDER BY: Ordina i risultati finali in base a uno/più campi.

4) **Descrivi, immaginando uno scenario a te familiare, il concetto di group by. Utilizza l'approccio che ritieni più efficiente per trasmettere il concetto (suggerimento: disegna anche una sola tabella in Excel o in word con poche colonne e pochi record e descrivi, basandosi sulla tabella stessa, un esempio di group by).**

Immaginando di avere una tabella con i dati delle vendite, con colonne come "Prodotto", "Data", "Quantità" e "Prezzo".
Utilizzando **GROUP BY** possiamo aggregare i dati per prodotto, calcolando la quantità totale venduta e il ricavo totale per ciascun prodotto.

```
SELECT Prodotto
, SUM(Quantità) AS Totale_Quantità
FROM Vendite
GROUP BY Prodotto;
```

5) **Descrivi la differenza tra uno schema OLTP e uno schema OLAP.**

La differenza sostanziale tra i due schemi è che il primo (OLTP) ha come obiettivo quello di ottimizzare le transazioni online, mentre il secondo (OLAP) ha quello di recuperare e analizzare i dati.

6) **Dato un medesimo scenario di analisi, qual è la differenza in termini di risultato ottenibile tra una join e una subquery?**

Con il Join riusciamo a combinare le righe di 2 o più tabelle in base ad una condizione da noi scelta.
Con una subquery non abbiamo altro che una query annidata all'interno di un'altra.
Il risultato restituito da quella annidata verrà poi utilizzata dalla query "esterna".

7) **Cosa si intende per DML e DDL?**

Con DML si intendono i comandi utilizzati per manipolare i dati del database.
(INSERT, UPDATE, DELETE).

Con DDL invece per definire e gestire la struttura del database.
(CREATE, ALTER, DROP).

8) **Quali istruzioni possono utilizzare per estrarre l'anno da un campo data? Proponi degli esempi.**

```
SELECT YEAR(data_colonna) AS Anno
FROM Tabella;
```

9) **Qual è la differenza tra gli operatori logici AND e OR?**

Con l'operatore AND entrambi le condizioni devono essere vere, mentre con l'operatore OR almeno una delle due deve essere vera.

10) **È possibile innestare una query nella clausola SELECT?**

Si è possibile innestare una query nella clausola SELECT.

11) **Qual è la differenza tra l'operatore logico OR e l'operatore logico IN?**

L'operatore OR viene utilizzato per specificare più di una condizione, dove almeno una di essere deve essere vera.

Con l'operatore IN invece si va a specificare se un valore corrisponde ad uno qualsiasi contenuto in un elenco.

12) **L'operatore logico BETWEEN include anche gli estremi del range specificato?**

Si, l’operatore BETWEEN include gli estremi del range specificato.

Case Study

Tabelle:

Product

- **ProductID (PK)**
- **ProductName**
- **CategoryID (FK)**

Category

- **CategoryID (PK)**
- **CategoryName**

Region

- **RegionID (PK)**
- **RegionName**
- **CountryID (FK)**

Country

- **CountryID (PK)**
- **CountryName**

Sales

- **SalesID (PK)**
- **ProductID (FK)**
- **RegionID (FK)**
- **SalesDate**
- **QuantitySold**

Schema del Database e relazioni

Product

ProductID | ProductName | CategoryID

Category

CategoryID | CategoryName

Region

RegionID | RegionName | CountryID

Country

CountryID | CountryName

Sales

SalesID | ProductID | RegionID | SalesDate | QuantitySold

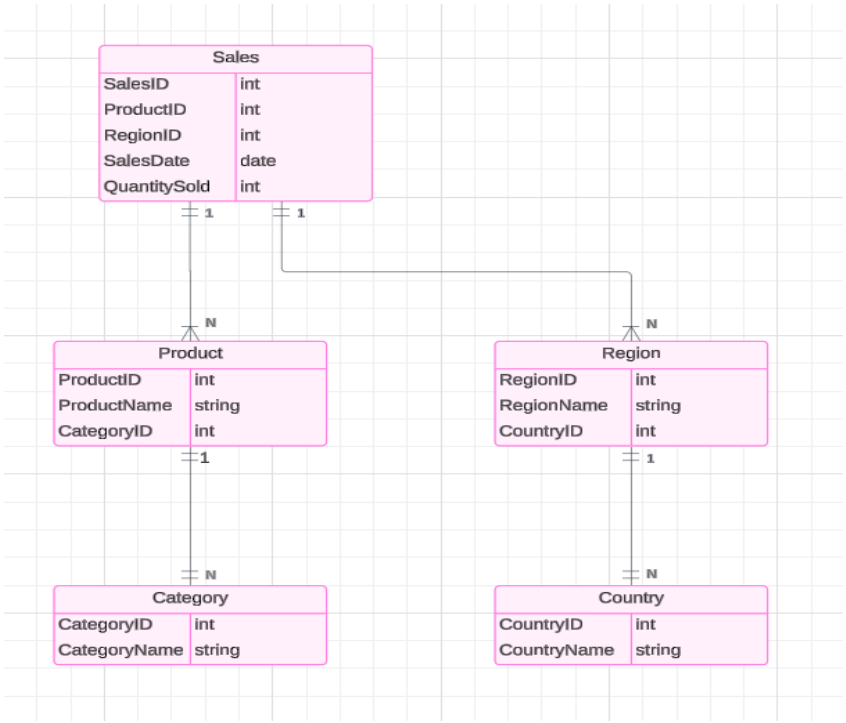
****

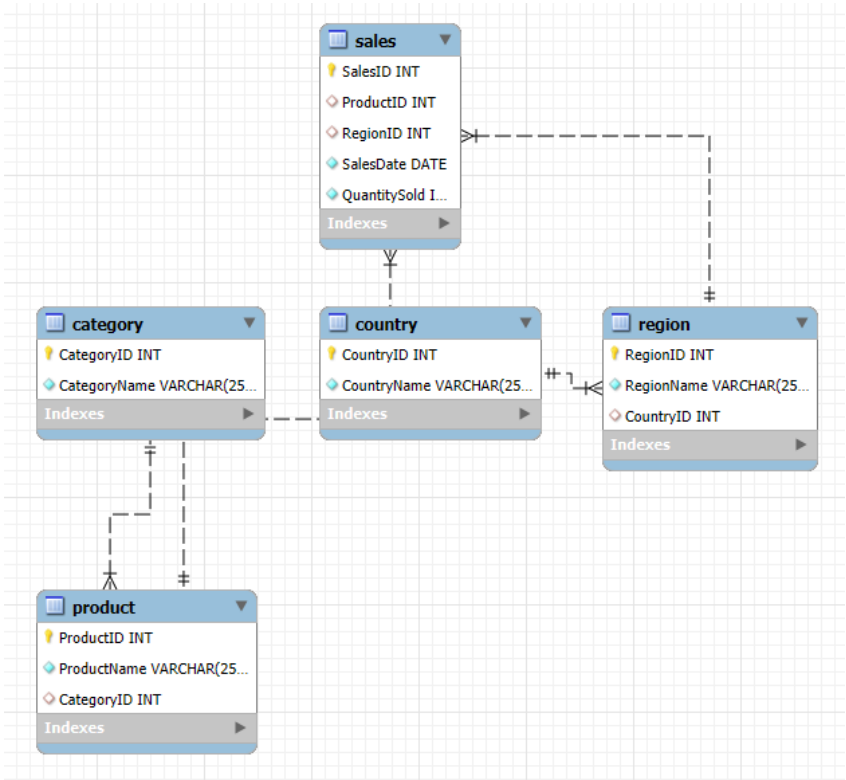
La tabella Product è collegata a Category tramite CategoryID (PK).

La tabella Region è collegato a Country tramite CountryID (PK).

La tabella Sales è collegato a Product tramite ProductID (FK) e a Region tramite RegionID (FK).

Progettazione Concettuale/Logica:





\\

Task 2: Descrivi la struttura delle tabelle che reputi utili e sufficienti a modellare lo scenario proposto tramite la sintassi DDL. Implementa fisicamente le tabelle utilizzando il DBMS SQL Server(o altro).

```
-- Creazione della tabella Category
CREATE TABLE Category (
   CategoryID INT PRIMARY KEY,
   CategoryName VARCHAR(255) NOT NULL
);

-- Creazione della tabella Country
CREATE TABLE Country (
   CountryID INT PRIMARY KEY,
   CountryName VARCHAR(255) NOT NULL
);

-- Creazione della tabella Product
CREATE TABLE Product (
   ProductID INT PRIMARY KEY,
   ProductName VARCHAR(255) NOT NULL,
   CategoryID INT,
   FOREIGN KEY (CategoryID) REFERENCES Category(CategoryID)
);

-- Creazione della tabella Region
CREATE TABLE Region (
   RegionID INT PRIMARY KEY,
   RegionName VARCHAR(255) NOT NULL,
   CountryID INT,
   FOREIGN KEY (CountryID) REFERENCES Country(CountryID)
);

-- Creazione della tabella Sales
CREATE TABLE Sales (
   SalesID INT PRIMARY KEY,
   ProductID INT,
   RegionID INT,
   SalesDate DATE NOT NULL,
   QuantitySold INT NOT NULL,
   FOREIGN KEY (ProductID) REFERENCES Product(ProductID),
   FOREIGN KEY (RegionID) REFERENCES Region(RegionID)
);
```

Task 3: Popola le tabelle utilizzando dati a tua discrezione (sono sufficienti pochi record per tabella; riporta le query utilizzate)

- Tabella Category
  INSERT INTO Category (CategoryID, CategoryName) **VALUES**
  (1, 'Biciclette'),
  (2, 'Giochi Da Tavolo'),
  (3, ‘Giocattoli’);

- Tabella Country
  INSERT INTO Country (CountryID, CountryName) **VALUES**
  (1, 'Francia'),
  (2, 'Germania'),
  (3, ‘Italia’);

- Tabella Product
  INSERT INTO Product (ProductID, ProductName, CategoryID) **VALUES**
  (1, 'Bici-100', 1),
  (2, ‘Bici-200', 1),
  (3, ‘Guanti Bici M', 2),
  (4, 'Guanti Bici L', 2);

- Tabella Region
  INSERT INTO Region (RegionID, RegionName, CountryID) **VALUES**
  (1, 'WestEurope', 1),
  (2, 'WestEurope', 2),
  (3, 'SouthEurope', 3),
  (4, 'SouthEurope', 4);

- Tabella Sales
  INSERT INTO Sales (SalesID, ProductID, RegionID, SalesDate, QuantitySold) **VALUES**
  (1, 1, 1, '2024-01-01', 10),
  (2, 2, 1, '2024-02-15', 5),
  (3, 3, 2, '2024-03-10', 8),
  (4, 4, 3, '2024-04-05', 12);

Task 4: Dopo aver popolate le tabelle, scrivi delle query utili a:

2) **Verificare l'univocità dei campi definiti come PK**:

```
SELECT column_name, COUNT(*)
FROM table_name
GROUP BY column_name
HAVING COUNT(*) > 1;
```

*Column_name e table_name andranno sostituiti con i nomi delle tabelle e delle colonne PK.

3) **Elenco delle transazioni con condizione sui giorni passati dalla data vendita**:

```
SELECT s.SalesID AS CodiceDocumento, s.SalesDate AS Data, p.ProductName AS NomeProdotto, c.CategoryName AS CategoriaProdotto, r.RegionName AS NomeRegione, r.CountryName AS NomeStato,

    DATEDIFF(CURDATE(), s.SalesDate) > 180 AS PiùDi180Giorni

FROM Sales s

JOIN Product p ON s.ProductID = p.ProductID
JOIN Category c ON p.CategoryID = c.CategoryID
JOIN Region r ON s.RegionID = r.RegionID
JOIN Country co ON r.CountryID = co.CountryID;
```

4) **Elenco dei prodotti con vendite superiori alla media delle vendite dell'ultimo anno**:

```
SELECT p.ProductID, SUM(s.QuantitySold) AS TotaleVenduto
FROM Sales s
JOIN Product p ON s.ProductID = p.ProductID
WHERE s.SalesDate BETWEEN DATE_SUB(CURDATE(), INTERVAL 1 YEAR) AND CURDATE()
GROUP BY p.ProductID
HAVING SUM(s.QuantitySold) > (
  SELECT AVG(TotaleVenduto)
  FROM (
    SELECT SUM(s.QuantitySold) AS TotaleVenduto
    FROM Sales s
    WHERE s.SalesDate BETWEEN DATE_SUB(CURDATE(), INTERVAL 1 YEAR) AND CURDATE()
    GROUP BY s.ProductID
  ) AS SubQuery
);
```

5) **Elenco dei prodotti venduti con fatturato totale per anno**:

```
SELECT p.ProductID, YEAR(s.SalesDate) AS Anno, SUM(s.QuantitySold * p.Price) AS FatturatoTotale
FROM Sales s
JOIN Product p ON s.ProductID = p.ProductID
GROUP BY p.ProductID, YEAR(s.SalesDate);
```

6) **Fatturato totale per stato per anno**:

```
SELECT co.CountryName AS NomeStato, YEAR(s.SalesDate) AS Anno, SUM(s.QuantitySold * p.Price) AS
FatturatoTotale
FROM Sales s
JOIN Product p ON s.ProductID = p.ProductID
JOIN Region r ON s.RegionID = r.RegionID
JOIN Country co ON r.CountryID = co.CountryID
GROUP BY co.CountryName, YEAR(s.SalesDate)
ORDER BY YEAR(s.SalesDate), FatturatoTotale DESC;
```

7) **Categoria di articoli maggiormente richiesta**:

```
SELECT c.CategoryName, SUM(s.QuantitySold) AS TotaleVenduto
FROM Sales s
JOIN Product p ON s.ProductID = p.ProductID
JOIN Category c ON p.CategoryID = c.CategoryID
GROUP BY c.CategoryName
ORDER BY TotaleVenduto DESC
LIMIT 1;
```

8) **Prodotti invenduti**:

```
SELECT p.ProductID, p.ProductName
FROM Product p
LEFT JOIN Sales s ON p.ProductID = s.ProductID
WHERE s.SalesID IS NULL;

\\

SELECT p.ProductID, p.ProductName
FROM Product p
JOIN Sales s ON p.ProductID = s.ProductID
GROUP BY p.ProductID, p.ProductName
HAVING SUM(s.QuantitySold) = 0;
```

9) **Creare una vista denormalizzata dei prodotti**:

```
CREATE VIEW ProductView AS
SELECT p.ProductID, p.ProductName, c.CategoryName
FROM Product p
JOIN Category c ON p.CategoryID = c.CategoryID;
```

10) **Creare una vista per le informazioni geografiche**:

```
CREATE VIEW GeoInfo AS
SELECT r.RegionID, r.RegionName, co.CountryName
FROM Region r
JOIN Country co ON r.CountryID = co.CountryID;
```