**Questions**

1. Cosa si intende per database?
2. Cos’è un DBMS?
3. Indica le principali clausole di uno statement SELECT in ordine di esecuzione logica. Descrivi per ciascuna delle clausole indicate la logica di funzionamento.
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).
5. Descrivi la differenza tra uno schema OLTP e uno schema OLAP.
6. Dato un medesimo scenario di analisi, qual è la differenza in termini di risultato ottenibile tra una join e una subquery?
7. Cosa si intende per DML e DDL?
8. Quali istruzioni possono utilizzare per estrarre l’anno da un campo data? Proponi degli esempi.
9. Qual è la differenza tra gli operatori logici AND e OR?
10. È possibile innestare una query nella clausola SELECT?
11. Qual è la differenza tra l’operatore logico OR e l’operatore logico IN?
12. L’operatore logico BETWEEN include anche gli estremi del range specificato?
13. Un database (DB) rappresenta un insieme strutturato di file che raggruppano i dati elaborati nelle applicazioni informatiche di una persona, di un gruppo di persone, di un’impresa, di istituzioni ecc. Un database può essere definito come una collezione di dati interdipendenti, insieme alla descrizione dei dati e delle relazioni tra di essi, oppure come una collezione di dati utilizzata in un’organizzazione, una collezione che è automatizzata, condivisa, definita rigorosamente in modo logico e coerente e controllata a livello centrale.
14. Un DBMS – Database Management System – e un istrumento software creato per facilitare la creazione e l’aggiornamento del database, l’interrogazione e gestione del database.
15. Lo statement SELECT presenta tre clausole principali: SELECT, FROM e WHERE:

* SELECT viene utilizzata per specificare quali attributi (colonne) devono apparire nel risultato della query;
* FROM e quella che consente di identificare/enumerare le relazioni dalle quali verranno estratte le informazioni relative alla consultazione;
* Tramite WHERE si stabilisce la condizione, che può essere semplice oppure complessa, che le righe delle tabelle (elencate nella clausola FROM) devono soddisfare per essere estratte nel risultato.

In generale, una consultazione semplice in SQL e rappresentata come segue:

SELECT C1, C2, …, Cn FROM R1, R2, …, Rm WHERE p.

L’esecuzione di una frase SELECT si concretizza nell’ottenimento di un risultato sotto forma di una tabella (relazione). Quando la clausola WHERE e assente, si considera implicitamente che p abbia il valore logico “vero”. Se appare il simbolo “\*” tra SELECT e FROM (al posto di C1, C2,…, Cn), nella tabella risultato verranno incluse tutte le colonne (attributi) da tutte le relazioni specificate nella clausola FROM. Inoltre, nella tabella risultato, non e obbligatoria che gli attributi abbiano lo stesso nome di quelli della tabella enumerata nella clausola FROM. Il cambiamento di realizza tramite l’opzione AS.

1. Il risultato di una frase SELECT che contiene la clausola GROUP BY e una tabella che verrà ottenuta raggruppando tutte le righe delle tabelle elencate nel FROM, che presentano lo stesso valore per una colonna o un gruppo di colonne.

Immagina di voler tenere traccia delle spese familiari in un mese. Abbiamo una tabella con i seguenti dati:

Tabella 1 SpeseFamiliari

|  |  |  |  |
| --- | --- | --- | --- |
| ID spesa | Data | Categoria | Importo |
| 1 | 01/02/2025 | Trasporti | 72,00 |
| 2 | 02/02/2025 | Alimentari | 126,00 |
| 3 | 03/02/2025 | Utenze | 265,00 |
| 4 | 05/02/2025 | Trasporti | 20,00 |
| 5 | 09/02/2025 | Alimentari | 186,00 |
| 6 | 10/02/2025 | Trasporti | 5,00 |
| 7 | 11/02/2025 | Alimentari | 85,00 |

Abbiamo come obiettivo di calcolare la somma delle spese per la categoria Alimentari.

In questo caso, utilizziamo GROUP BY per raggruppare le spese per categoria e visualizzare la somma totale per ciascuna categoria.

La query in MySql:

*SELECT Categoria, SUM(Importo) AS TOTALE*

*FROM SpeseFamiliari*

*Where Categoria = ‘Alimentari’*

*GROUP BY Categoria;*

Tabella 2 Risultato

|  |  |
| --- | --- |
| Categoria | TOTALE |
| Alimentari | 397,00 |

1. Uno schema OLAP (Online Analytical Processing) combina e raggruppa i dati, consentendo di analizzarli da prospettive diverse. Al contrario, l’OLTP memorizza e aggiorna grandi volumi di dati transazionali in modo efficiente e affidabile. I database OLTP possono rappresentare una delle varie origi dati per un sistema OLAP.
2. Una JOIN viene utilizzata quando **vogliamo unire e visualizzare dati da più tabelle in un unico risultato.**

Invece, la subquery la utilizziamo quando **vogliamo filtrare, calcolare o estrarre dati specifici** prima di eseguire la query principale.

1. DML – Data Manipulation Language – consente di gestire e manipolare i dati all’interno delle tabelle esistenti (senza alterare la struttura del database). Ad esempio i comandi DELETE FROM…, INSERT INTO.., ecc.

DDL – Data Definition Language – consente di definire e modificare la struttura del database. Ad esempio i comandi CREATE TABLE…, DROP TABLE…, ecc.

1. La funzione **YEAR (*date\_value*).**  Ad esempio, utilizzando la Tabella 1 SpeseFamiliari :

*SELECT YEAR(data) AS Anno FROM SpeseFamiliari;*

Ritornerà

A screenshot of a computer

AI-generated content may be incorrect.

1. L’operatore AND restituisce TRUE solo se tutte le condizioni sono vere mentre l’operatore OR restituisce TRUE se almeno una delle condizioni e vera.
2. Si, e possibile innestare una query nella clausola SELECT utilizzando una **subquery**.

Ad esempio, utilizzando la Tabella1 SpeseFamiliari, supponiamo che vogliamo ottenere una collona aggiuntiva con l’elenco delle spese, con il totale delle spese per ciascuna categoria.

*SELECT IDspesa, Data, Categoria, Importo,*

*(SELECT SUM(Importo)*

*FROM SpeseFamiliari o2*

*WHERE o2.Categoria = o1.Categoria) AS TotaleCategoria*

*FROM SpeseFamiliari o1;*

A screenshot of a computer

AI-generated content may be incorrect.

1. L’operatore OR restituisce TRUE se almeno una delle condizioni e vera mentre l’operatore IN restituisce TRUE se l’operatore e uguale a una delle espressioni elencate.
2. Si, l’operatore logico BETWEEN include anche gli estremi del range specificato. Ad esempio:

*SELECT \**

*FROM SpeseFamiliari*

*WHERE Importo BETWEEN 5 AND 265;*



A screenshot of a computer

AI-generated content may be incorrect.

**Case Study**

ToysGroup è un’azienda che distribuisce articoli (giocatoli) in diverse aree geografiche del mondo.

I prodotti sono classificati in categorie e i mercati di riferimento dell’azienda sono classificati in regioni di vendita.

In particolare:

1. Le entità individuabili in questo scenario sono le seguenti:

* Product
* Region
* Sales

1. Le relazioni tra le entità possono essere descritte nel modo seguente:

* Product e Sales
* Un prodotto puo’ essere venduto tante volte (o nessuna) per cui è contenuto in una o più transazioni di vendita.
* Ciascuna transazione di vendita è riferita ad uno solo prodotto
* Region e Sales
* Possono esserci molte o nessuna transazione per ciascuna regione
* Ciascuna transazione di vendita è riferita ad una sola regione

1. Le entità Product e Region presentano delle gerarchie:

* L’entità prodotto contiene, oltre alle informazioni del singolo prodotto, anche la descrizione della categoria di appartenenza. L’entità prodotto contiene quindi una gerarchia: un prodotto puo’ appartenere ad una sola categoria mentre la stessa categoria puo’ essere associata a molti prodotti diversi.

*Esempio: gli articoli ‘Bikes-100’ e ‘Bikes-200’ appartengono alla categoria Bikes; gli articoli ‘Bike Glove M’ e ‘Bike Gloves L’ sono classificati come Clothing.*

* L’entità regione contiene una gerarchia: più stati sono classificati in una stessa regione di vendita e una stessa regione di vendita include molti stati.

*Esempio: gli stati ‘France’ e ‘Germany’ sono classificati nella region WestEurope; gli stati ‘Italy’ e ‘Greece’ sono classificati nel mercato SouthEurope*.

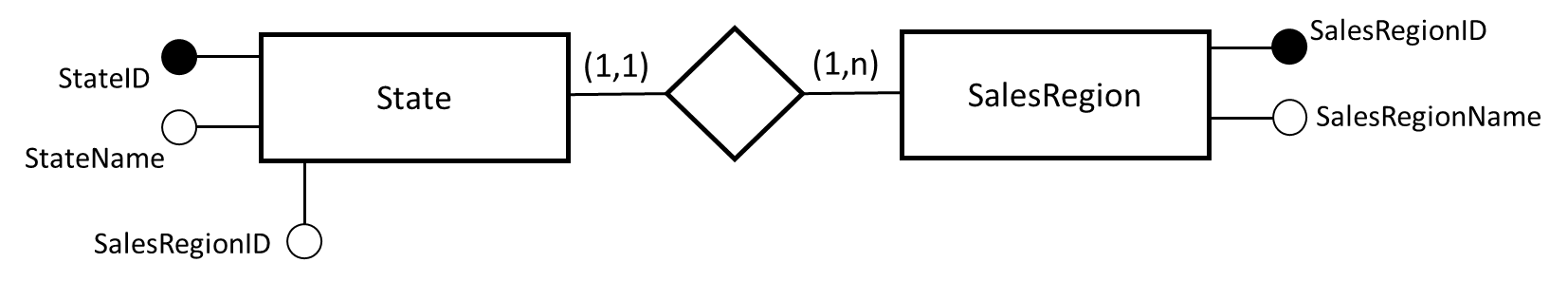
È necessario progettare e implementare fisicamente un database che modelli lo scenario garantendo l’**integrità referenziale** e la **minimizzazione della ridondanza dei dati.**

In altre parole, progetta opportunamente un numero di tabelle e di relazioni tra queste sufficiente a garantire la **consistenza del dato.**

**Task 1: Proponi una progettazione concettuale e logica della base dati**

La progettazione concettuale deve includere tutte le entità coinvolte e le relazioni tra queste. Per ciascuna entità indica l’attributo chiave e i principali attributi descrittivi (non è necessario indicare tutti gli attributi).

*Esempio di schema E/R*



*Lo schema proposto è puramente esemplificativo e non esaustivo o completo per la soluzione!*

La progettazione logica deve includere, per ciascuna tabella, tutte le colonne che poi verranno implementate fisicamente e deve esplicitare la cardinalità dei campi utilizzati per definire la relazione.

*Esempio di schema grafico delle tabelle e delle relazioni tra le stesse.*



n

1



*Il diagramma è puramente esemplificativo e non esaustivo o completo per la soluzione!*

**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).**

CREATE TABLE table\_name (

column1 datatype option,

column2 datatype option,

column3 datatype,

....

);

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

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

1. Verificare che i campi definiti come PK siano univoci. In altre parole, scrivi una query per determinare l’univocità dei valori di ciascuna PK (una query per tabella implementata).
2. Esporre l’elenco delle transazioni indicando nel result set il codice documento, la data, il nome del prodotto, la categoria del prodotto, il nome dello stato, il nome della regione di vendita e un campo booleano valorizzato in base alla condizione che siano passati più di 180 giorni dalla data vendita o meno (>180 -> True, <= 180 -> False)
3. Esporre l’elenco dei prodotti che hanno venduto, in totale, una quantità maggiore della media delle vendite realizzate nell’ultimo anno censito. (ogni valore della condizione deve risultare da una query e non deve essere inserito a mano). Nel result set devono comparire solo il codice prodotto e il totale venduto.
4. Esporre l’elenco dei soli prodotti venduti e per ognuno di questi il fatturato totale per anno.
5. Esporre il fatturato totale per stato per anno. Ordina il risultato per data e per fatturato decrescente.
6. Rispondere alla seguente domanda: qual è la categoria di articoli maggiormente richiesta dal mercato?
7. Rispondere alla seguente domanda: quali sono i prodotti invenduti? Proponi due approcci risolutivi differenti.
8. Creare una vista sui prodotti in modo tale da esporre una “versione denormalizzata” delle informazioni utili (codice prodotto, nome prodotto, nome categoria)
9. Creare una vista per le informazioni geografiche

**TASK 1:**

A diagram of a diagram

AI-generated content may be incorrect.

Figura 1 progettazione concettuale (uso app,diagrams.net)

A screenshot of a computer

AI-generated content may be incorrect.

Figura 2 Schema grafico delle tabelle e delle relazioni tra le stesse

A screenshot of a computer

AI-generated content may be incorrect.

Figura 3 EER Diagram

L'entità **DocInfo** è stata introdotta per garantire una chiara separazione tra le transazioni di vendita e i dettagli della vendita. Senza questa entità, sarebbe stato necessario memorizzare informazioni come lo Stato e il totale pagato direttamente nella tabella **Sales**, portando a ridondanza dei dati e violazione dei principi di normalizzazione. Grazie a **Documento**:

* Mantengo la relazione **1:N tra Region e State**, rispettando la gerarchia originale.
* Ogni vendita in **Sales** è associata a un unico documento, che contiene il riferimento a uno **Stato**, evitando di modificare direttamente la relazione tra **Sales e State**.
* È possibile ottenere tutte le informazioni relative a una vendita (Stato, Regione, totale pagato, dettagli aggiuntivi) senza duplicare dati.

**TASK 2:**

CREATE TABLE Category (

CategoryID INT AUTO\_INCREMENT PRIMARY KEY,

CategoryName varchar (50) NOT NULL,

Description TEXT

);

CREATE TABLE Product (

ProductID INT PRIMARY KEY,

ProductName VARCHAR (50) NOT NULL,

CategoryID INT,

Price decimal (10,2) NOT NULL,

Description TEXT,

FOREIGN KEY (CategoryID) references Category (CategoryID)

);

CREATE TABLE Region (

RegionID INT PRIMARY KEY,

RegionName varchar (50) NOT NULL,

Description TEXT

);

CREATE TABLE State (

StateID INT PRIMARY KEY,

StateName VARCHAR(255) NOT NULL,

RegionID INT,

CONSTRAINT FK\_State\_Region

FOREIGN KEY (RegionID) REFERENCES Region(RegionID)

);

CREATE TABLE DocInfo (

DocNo VARCHAR (50) PRIMARY KEY,

StateID INT,

Details VARCHAR (255),

FOREIGN KEY (StateID) REFERENCES State(StateID)

);

CREATE TABLE Sales (

SalesID INT AUTO\_INCREMENT PRIMARY KEY,

ProductID INT,

RegionID INT,

QuantitySold INT NOT NULL,

DocNo Varchar (15),

SaleDate DATE NOT NULL,

TotalAmount DECIMAL(10,2) NOT NULL,

FOREIGN KEY (ProductID) REFERENCES Product(ProductID),

FOREIGN KEY (RegionID) REFERENCES State(RegionID),

FOREIGN KEY (DocNo) REFERENCES DocInfo (DocNo)

);

**TASK 3:**

INSERT INTO Category (CategoryName, Description)

VALUES

('Bikes', 'All types of bicycles'),

('Clothing', 'Apparel for cycling'),

('Accessories', 'Additional gear for bikes'),

('Toys', 'Various children’s toys'),

('Electronics', 'Gadgets and devices'),

('Furniture', 'Home and office furniture'),

('Sports', 'Sports equipment and gear'),

('Books', 'Various types of books'),

('Music', 'Instruments and accessories'),

('Outdoor', 'Camping and outdoor gear');

INSERT INTO Product (ProductID, ProductName, CategoryID, Price, Description)

VALUES

(651485, 'Mountain Bike', 1, 500.00, 'A rugged mountain bike'),

(728547, 'Road Bike', 1, 700.00, 'A lightweight road bike'),

(125485, 'Bike Helmet', 2, 50.00, 'Safety helmet for biking'),

(986574, 'Cycling Gloves', 2, 20.00, 'Protective gloves for cyclists'),

(777458, 'Bike Lock', 3, 30.00, 'Heavy-duty bike lock'),

(654, 'Action Figure', 4, 25.00, 'Superhero action figure'),

(98450, 'RC Car', 4, 100.00, 'Remote controlled car'),

(999458, 'Smartwatch', 5, 200.00, 'Wearable fitness tracker'),

(131354, 'Office Chair', 6, 150.00, 'Ergonomic office chair'),

(248574, 'Camping Tent', 10, 120.00, 'Waterproof camping tent');

INSERT INTO Region (RegionID, RegionName, Description)

VALUES

(1, 'West Europe', 'Western European countries'),

(2, 'South Europe', 'Southern European countries'),

(3, 'North Europe', 'Northern European countries'),

(4, 'East Europe', 'Eastern European countries');

INSERT INTO State (StateID, StateName, RegionID)

VALUES

(45, 'France', 1),

(6, 'Germany', 1),

(25, 'United Kingdom', 1),

(28, 'Spain', 2),

(1, 'Italy', 2),

(35, 'Greece', 2),

(17, 'Sweden', 3),

(22, 'Norway', 3),

(29, 'Poland', 3),

(2, 'Romania', 4),

(32, 'Russia', 4);

INSERT INTO DocInfo (DocNo, StateID, Details)

VALUES

('DOC001', 25, 'Sale completed for 2500.00 EUR'),

('DOC002', 6, 'Sale processed for 2100.00 EUR'),

('DOC003', 25, 'Payment received: 500.00 EUR'),

('DOC004', 28, 'Paid by customer: 300.00 EUR'),

('DOC005', 1, 'Invoice settled: 210.00 EUR'),

('DOC006', 35, 'Finalized transaction of 450.00 EUR'),

('DOC007', 17, 'Processed payment: 600.00 EUR'),

('DOC008', 2, 'High-value transaction: 4000.00 EUR'),

('DOC009', 22, 'Client purchase of 800.00 EUR'),

('DOC010', 32, 'Confirmed order for 720.00 EUR'),

('DOC123', 35, 'Sale completed for 2500.00 EUR'),

('DOC124', 29, 'Sale processed for 1500.00 EUR'),

('DOC125', 1, 'Payment received: 500.00 EUR'),

('DOC126', 2, 'High-value transaction: 9000.00 EUR'),

('DOC127', 25, 'Client purchase of 1200.00 EUR'),

('DOC128', 32, 'Large order finalized: 10500.00 EUR');

INSERT INTO Sales (ProductID, RegionID, QuantitySold, DocNo, SaleDate, TotalAmount)

VALUES

(651485, 1, 5, 'DOC001', '2018-02-15', 2500.00),

(728547, 1, 3, 'DOC002', '2019-06-25', 2100.00),

(125485, 1, 10, 'DOC003', '2020-08-14', 500.00),

(986574, 2, 15, 'DOC004', '2021-03-22', 300.00),

(777458, 2, 7, 'DOC005', '2022-07-19', 210.00),

(654, 2, 9, 'DOC006', '2023-01-30', 450.00),

(98450, 3, 12, 'DOC007', '2019-11-10', 600.00),

(999458, 4, 20, 'DOC008', '2020-12-05', 4000.00),

(131354, 3, 4, 'DOC009', '2021-09-28', 800.00),

(248574, 4, 6, 'DOC010', '2022-05-16', 720.00),

(651485, 2, 5, 'DOC123', '2025-02-24', 2500.00),

(125485, 3, 30, 'DOC124', '2025-02-23', 1500.00),

(654, 2, 20, 'DOC125', '2025-02-22', 500.00),

(131354, 4, 60, 'DOC126', '2025-02-21', 9000.00),

(777458, 1, 40, 'DOC127', '2025-02-20', 1200.00),

(131354, 4, 70, 'DOC128', '2025-02-19', 10500.00);

**TASK 4:**

1. Verificare che i campi definiti come PK siano univoci. In altre parole, scrivi una query per determinare l’univocità dei valori di ciascuna PK (una query per tabella implementata).

A screenshot of a computer

AI-generated content may be incorrect.

**Tabella Product:**

SELECT ProductID, COUNT(\*)

FROM Product

GROUP BY ProductID

HAVING COUNT(\*) > 1;

Questa query conta quante volte ogni valore della chiave primaria appare nella tabella Product e restituisce solo le righe dove un valore appare più di una volta. In questo caso non restituisce nulla e dimostra che i campi definiti come PK nella tabella Product sono univoci.

**Tabella Category:**

Utilizzerò un'altra query e mi aiuterò delle subquery utilizzando DISTINCT per vedere se la chiave primaria e univoca:

SELECT

(SELECT COUNT(DISTINCT CategoryID) FROM Category) AS distinct\_values,

(SELECT COUNT(\*) FROM Category) AS total\_rows;

**A screenshot of a computer screen

AI-generated content may be incorrect.**

**Tabella Region:**

SELECT COUNT(DISTINCT RegionID) AS distinct\_count, COUNT(\*) AS total\_count

FROM Region;

A screenshot of a computer

AI-generated content may be incorrect.

Questa query confronta il numero dei valori distinti con il numero totale di righe nella tabella. I due numeri coincidono quindi dimostra l’univocità.

**Tabella State:**

SELECT t1.StateID, Count(\*)

FROM State t1

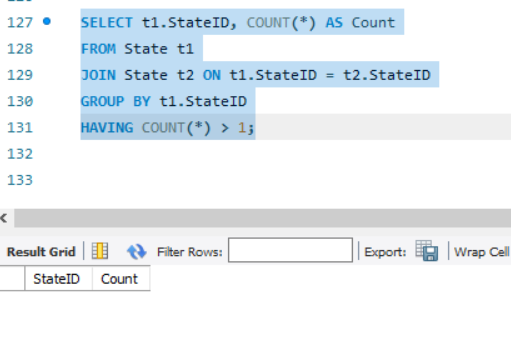
JOIN State t2 ON t1.StateID = t2.StateID

Group BY t1.StateID

Having Count(\*) > 1;

Con questa query unisco la tabella con se stessa, raggruppando i risultati per StateID e filtrando solo quelli che appaiano più di una volta. Se la query restituisse dei risultati, significherebbe che esistono righe con lo stesso valore e quindi ci sono duplicati.

In questo caso non vengono restituiti risultati quindi la chiave primaria e univoca.



**Tabella Sales:**

SELECT SalesID, COUNT(\*)

FROM Sales

GROUP BY SalesID

HAVING COUNT(\*) > 1;

A screenshot of a computer

AI-generated content may be incorrect.

*2)* Esporre l’elenco delle transazioni indicando nel result set il codice documento, la data, il nome del prodotto, la categoria del prodotto, il nome dello stato, il nome della regione di vendita e un campo booleano valorizzato in base alla condizione che siano passati più di 180 giorni dalla data vendita o meno (>180 -> True, <= 180 -> False)

*SELECT*

*Sales.DocNo AS CodiceDocumento,*

*Sales.SaleDate AS Data,*

*Product.ProductName AS NomeProdotto,*

*Category.CategoryName AS CategoriaProdotto,*

*State.StateName AS NomeStato,*

*Region.RegionName AS NomeRegione,*

*(SELECT IF(DATEDIFF(CURDATE(), Sales.SaleDate) > 180,'TRUE', 'FALSE')) AS PiuDi180Giorni*

*FROM Sales*

*JOIN Product ON Sales.ProductID = Product.ProductID*

*JOIN Category ON Product.CategoryID = Category.CategoryID*

*JOIN Region ON Sales.RegionID = Region.RegionID*

*JOIN DocInfo ON Sales.DocNo = DocInfo.DocNo*

*JOIN State ON DocInfo.StateID = State.StateID;*

A screenshot of a computer

AI-generated content may be incorrect.

3) Esporre l’elenco dei prodotti che hanno venduto, in totale, una quantità maggiore della media delle vendite realizzate nell’ultimo anno censito. (ogni valore della condizione deve risultare da una query e non deve essere inserito a mano). Nel result set devono comparire solo il codice prodotto e il totale venduto.

*SELECT ProductID, SUM(QuantitySold) AS TotaleVenduto*

*FROM Sales*

*WHERE YEAR(Sales.SaleDate) = (*

*SELECT MAX(YEAR(SaleDate))*

*FROM Sales*

*)*

*GROUP BY ProductID*

*HAVING SUM(QuantitySold) > (*

*SELECT AVG(TotalePerProdotto)*

*FROM (*

*SELECT SUM(QuantitySold) AS TotalePerProdotto*

*FROM Sales*

*WHERE YEAR(SaleDate) = (*

*SELECT MAX(YEAR(SaleDate))*

*FROM Sales*

*)*

*GROUP BY ProductID*

*) AS MediaVendite*

*);*

*A screenshot of a computer

AI-generated content may be incorrect.*

4) Esporre l’elenco dei soli prodotti venduti e per ognuno di questi il fatturato totale per anno.

*SELECT*

*YEAR(Sales.SaleDate) AS Anno,*

*Sales.ProductID,*

*SUM(Sales.QuantitySold \* Product.Price) AS FatturatoTotale*

*FROM Sales*

*JOIN Product ON Sales.ProductID = Product.ProductID*

*WHERE Sales.QuantitySold > 0 -- Assicura che siano considerati solo prodotti effettivamente venduti*

*GROUP BY Anno, Sales.ProductID*

*ORDER BY Anno DESC, FatturatoTotale DESC;*

A screenshot of a computer

AI-generated content may be incorrect.

5) Esporre il fatturato totale per stato per anno. Ordina il risultato per data e per fatturato decrescente.

*SELECT*

*YEAR(Sales.SaleDate) AS Anno,*

*State.StateName,*

*SUM(Sales.QuantitySold \* Product.Price) AS FatturatoTotale*

*FROM Sales*

*JOIN DocInfo ON Sales.DocNo = DocInfo..DocNo*

*Join Product ON Sales.ProductID = Product.ProductID*

*JOIN State ON State.StateID = DocInfo.StateID*

*GROUP BY Anno, State.StateName*

*ORDER BY Anno ASC, FatturatoTotale DESC;*

A screenshot of a computer

AI-generated content may be incorrect.

6*)* Rispondere alla seguente domanda: qual è la categoria di articoli maggiormente richiesta dal mercato?

*-* Questa query calcola la categoria che ha venduto più pezzi:

*SELECT*

*Category.CategoryName,*

*SUM(Sales.QuantitySold) AS QuantitaTotaleVenduta*

*FROM Sales*

*JOIN Product ON Product.ProductID = Sales.ProductID*

*JOIN Category ON Product.CategoryID = Category.CategoryID*

*GROUP BY Category.CategoryName*

*ORDER BY QuantitaTotaleVenduta DESC*

*LIMIT 1;*

A screenshot of a computer

AI-generated content may be incorrect.

Invece, questa query ci restituisce la categoria che è stata acquistata più spesso:

*SELECT Category.CategoryName AS Categoria, COUNT(Sales.SalesID) AS NumeroDiTransazioni*

*FROM Sales*

*JOIN Product ON Sales.ProductID = Product.ProductID*

*JOIN Category ON product.CategoryID = category.CategoryID*

*GROUP BY category.CategoryName*

*ORDER BY NumeroDiTransazioni DESC*

*LIMIT 1;*

A screenshot of a computer

AI-generated content may be incorrect.

7) Rispondere alla seguente domanda: quali sono i prodotti invenduti? Proponi due approcci risolutivi differenti.

*SELECT ProductID, ProductName*

*FROM Product*

*WHERE ProductID NOT IN (SELECT ProductID FROM Sales);*

*A screenshot of a computer

AI-generated content may be incorrect.*

8) Creare una vista sui prodotti in modo tale da esporre una “versione denormalizzata” delle informazioni utili (codice prodotto, nome prodotto, nome categoria)

*CREATE VIEW ProductView AS*

*SELECT Product.ProductID AS ID, Product.ProductName AS Nome, Category.CategoryName AS Categoria*

*FROM Product*

*JOIN Category ON Product.CategoryID = Category.CategoryID;*

A screenshot of a computer

AI-generated content may be incorrect.

9) Creare una vista per le informazioni geografiche

*CREATE VIEW GeographyView AS*

*SELECT State.StateID AS ID, State.StateName AS Paese, Region.RegionName AS Regione*

*FROM State*

*JOIN Region ON state.RegionID = Region.RegionID;*

*A screenshot of a computer

AI-generated content may be incorrect.*