**Questions**

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

Con il termine database si intende una collezione di dati consistenti, organizzata in modo logico e coerente per facilitare la scrittura, la lettura, l’aggiornamento o le modifiche ai dati e l’eliminazione eventuale di dati (le cosiddette operazioni CRUD – create, read, update, e delete -).

1. **Cos’è un DBMS?**

Un DBMS (acronimo di Database Management System) è un sistema software di gestione di basi dati ed è anche detto gestore o motore del database. È progettato per consentire la creazione, la manipolazione e l’interrogazione efficiente di database. È ospitato su architettura hardware dedicata oppure su semplice pc.

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

Le principali clausole di uno Statement Select sono le seguenti:

1. SELECT 🡪 consente di selezionare/indicare i campi desiderati nel result set.
2. FROM 🡪 è la sorgente dati (tabella) che occorre interrogare
3. WHERE 🡪 è la condizione di ricerca, ovvero una clausola che consente di applicare un filtro sulle righe della tabella restituita dalla from.
4. GROUP BY 🡪 consente di raggruppare le righe della tabella filtrata per ogni combinazione univoca della group by list (cioè i campi rispetto i quali aggregare).
5. HAVING 🡪 è una condizione di ricerca che consente di applicare un filtro sui raggruppamenti restituiti dalla group by.
6. ORDER BY 🡪 consente di indicare un ordinamento (desc se decrescente, o asc – che può anche essere omesso – se crescente).
7. **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).**

Questa tabella, che chiamerò Tabelladip elenca gli impiegati di un’azienda e la loro mansione.

Tramite la group by andrò a mostrare un result set che mi dirà il totale del capitale mensile destinato a ogni singolo gruppo di impiegati (dunque i soldi totali destinati per il pagamento degli stipendi agli assistenti, i soldi totali destinati al pagamento degli stipendi dei responsabili e i soldi totali destinati al pagamento degli stipendi dei capi settore).

|  |  |  |  |
| --- | --- | --- | --- |
| ID\_IMPIEGATO | NOMINATIVO | MANSIONE | STIPENDIO |
| A1 | Simona | Assistente | 1350 |
| A2 | Maria | Assistente | 1200 |
| A3 | Erika | Assistente | 1400 |
| A4 | Pietro | Responsabile | 2000 |
| A5 | Andrea | Responsabile | 1900 |
| A6 | Susanna | Capo settore | 2500 |
| A7 | Giovanni | Capo settore | 2300 |

Il mio result set sarà il seguente:

|  |  |
| --- | --- |
| MANSIONE | STIPENDIO |
| Assistente | 3950 |
| Responsabile | 3900 |
| Capo settore | 4700 |

1. **Descrivi la differenza tra uno schema OLTP e uno schema OLAP.**

OLTP (acronimo di Online Transaction Processing) : il suo scopo principale è l’inserimento dati. Lo schema OLTP è un sistema che impacchetta i dati in più tabelle e dunque garantisce la consistenza del dato e l’integrità referenziale, è uno schema normalizzato.

OLAP (acronimo di On-Line Analytical Processing) è un tipo di elaborazione dati che consiste nel distribuire i dati su un’unica tabella, quindi il suo scopo non è l’inserimento dati ma la fruizione di essi ovvero il recupero e l’analisi, pertanto è la componente base del data warehouse, è uno schema altamente denormalizzato che ha ridondanza, ed è ottimale per l’analisi dati e per la costruzione di soluzioni BI.

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

Una join combina colonne da tabelle diverse basandosi su una condizione di correlazione specificata (ad esempio corrispondenza tra PK e FK) restituendo un set di risultati che include colonne da entrambe le tabelle coinvolte mentre una subquery è una query incorporata in un'altra query e restituisce un set di risultati o un singolo valore, che può essere utilizzato come parte di una condizione nella query esterna.

1. **Cosa si intende per DML e DDL?**  
   DDL (acronimo di data definition language) è un linguaggio utilizzato per manipolare i dati memorizzati nel database e include istruzioni come CREATE, ALTER, DROP, ADD.

DML (acronimo di data manipulation language) è un linguaggio utilizzato per definire e gestire la struttura del database è include istruzioni come SELECT, INSERT, UPDATE, DELETE.

1. **Quali istruzioni possono utilizzare per estrarre l’anno da un campo data? Proponi degli esempi.**

Per estrarre l’anno in un campo data si utilizza la funzione YEAR.   
Dato un campo data, ad esempio 1992-10-12 (YYYY-MM-DD) in una colonna detta DateBirth, tramite la funzione YEAR(DateBirth) vedremo nel result 1992 (solo l’anno).

In alternativa possiamo utilizzare la funzione DATE\_FORMAT, con la quale dato un campo data ad esempio ad esempio 1992-10-12 (YYYY-MM-DD) vado ad eseguire DATE\_FORMAT(DateBirth, "%Y") andrò ad avere nel result set 1992, ovvero la data formattata in formato anno a quattro cifre tramite il token %Y.

In ultimo è possibile estrarre tramite il token %y (minuscola) solo le ultime due cifre dell’anno.

Quindi eseguendo DATE\_FORMAT(DateBirth, “%y”) il result set sarà 92.

1. **Qual è la differenza tra gli operatori logici AND e OR?**

Gli operatori logici AND e OR sono utilizzati per combinare condizioni in un'espressione logica nelle query.

L’operatore AND restuisce come risultato TRUE se entrambe le condizioni sono vere. L’operatore OR restituisce TRUE se almeno una delle due condizioni è vera.

1. **È possibile innestare una query nella clausola SELECT?**

Si, è possibile innestare una query nella clausola SELECT utilizzando una subquery nella clausola SELECT stessa, il risultato sarà restituito come una nuova colonna, insieme alle altre colonne specificate nella select.

1. **Qual è la differenza tra l’operatore logico OR e l’operatore logico IN?**

L'operatore logico OR restituisce true se almeno una delle condizioni è vera e si utilizza quando si vuole selezionare le righe che soddisfano almeno una delle condizioni specificate, mentre l’operatore logico IN è utilizzato per verificare se un valore è uguale a uno qualsiasi dei valori specificati in un elenco e quindi si utilizza quando si vuole confrontare un valore con un insieme di valori specificati.

1. **L’operatore logico BETWEEN include anche gli estremi del range specificato?**

Si, li include. Ad esempio se cerco un valore compreso tra 10 e 20 l’operatore between mi restituirà tutti i valori compresi tra 10 e 20, 10 e 20 compresi.

1. **Che cos’è una SELF JOIN? Proponi un esempio.**

La self join consiste nel mettere in join più istanze della stessa tabella. È utile quando si desidera confrontare righe all'interno della stessa tabella basandosi su determinate condizioni.

Esempio:

SELECT

DC1.ClassKey

, CONCAT(DC1.FirstName, ' ', DC1.LastName) as Student

, DC1.ParentClassKey

, CONCAT(DC2.FirstName, ' ', DC2.LastName) AS Professor

, DC2.ClassKey

FROM dimclass AS DC1

JOIN dimclass AS DC2

ON DC1.ParentClassKey = DC2.ClassKey

WHERE DC1.Class\_Person\_Type = 1;

**Case Study**

ToysGroup è un’azienda che distribuisce articoli (giocattoli) 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*.

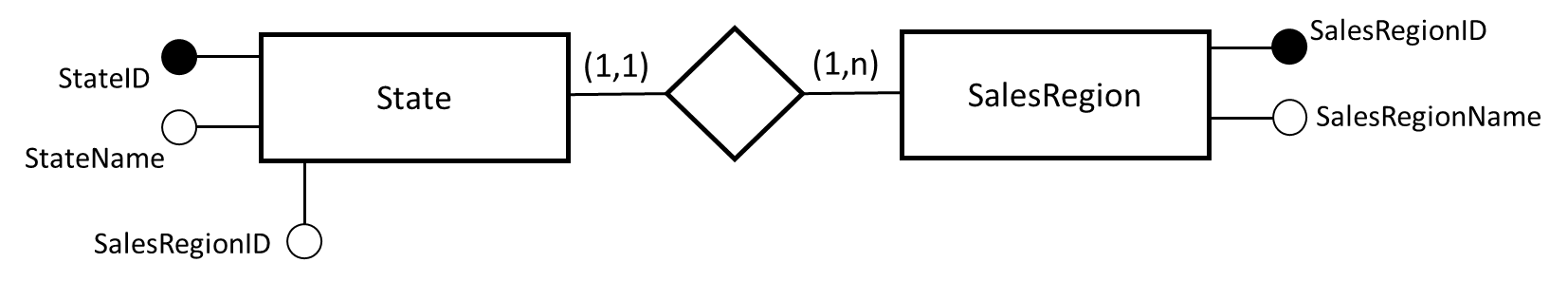
È necessario progettare e implementare fisicamente un database che modelli lo scenario garantendo l’**integrità referenziale** e la **minimizzazione della ridondanza dei dati.**

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 puram ente esemplificativo e non esaustivo o completo per la soluzione!*

La progettazione concettuale 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.*



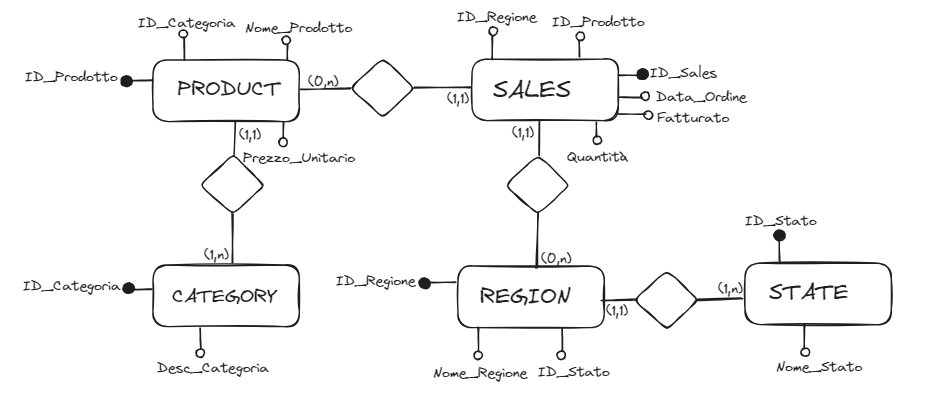
n

1

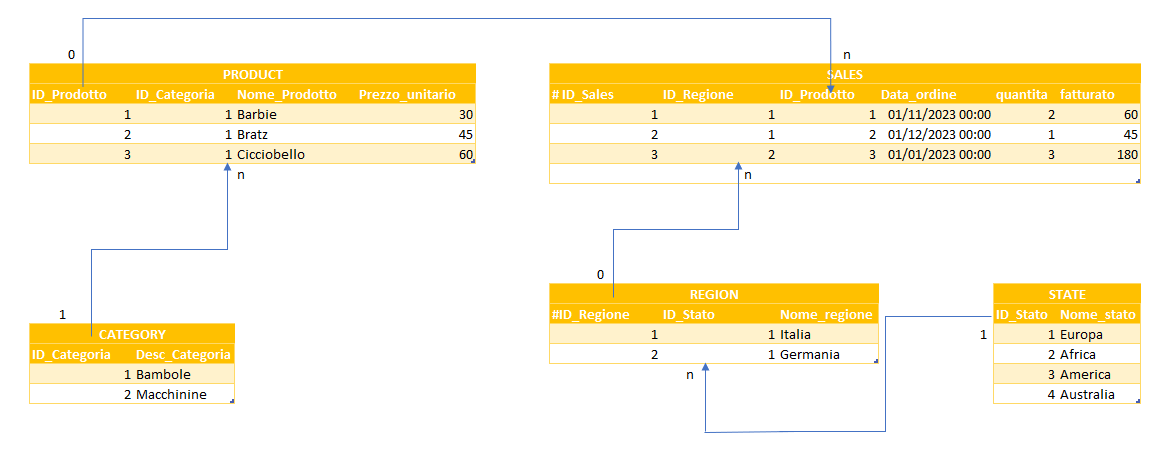


*Il diagramma è puramente esemplificativo e non esaustivo o completo per la soluzione!*

***Schema E/R Svolto***

**

***SCHEMA LOGICO SVOLTO***

**

**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 MySQL (o altro).**

CREATE TABLE table\_name (

column1 datatype option,

column2 datatype option,

column3 datatype,

....

);

CREATE TABLE Category (

ID\_Categoria INT,

Desc\_Categoria VARCHAR(32) NOT NULL,

CONSTRAINT PK\_Categoria PRIMARY KEY (ID\_Categoria)

);

CREATE TABLE Product (

ID\_Prodotto INT,

ID\_Categoria INT,

Nome\_Prodotto VARCHAR(32) NOT NULL,

Prezzo\_unitario INT NOT NULL,

CONSTRAINT PK\_Product PRIMARY KEY (ID\_Prodotto),

CONSTRAINT FK\_Categoria FOREIGN KEY (ID\_Categoria)

REFERENCES Category (ID\_Categoria)

);

CREATE TABLE State (

ID\_Stato INT,

Nome\_stato VARCHAR(32) NOT NULL,

CONSTRAINT PK\_State PRIMARY KEY (ID\_Stato)

);

CREATE TABLE Region (

ID\_Regione INT,

ID\_Stato INT,

Nome\_regione VARCHAR(32) NOT NULL,

CONSTRAINT PK\_Region PRIMARY KEY (ID\_Regione),

CONSTRAINT FK\_State FOREIGN KEY (ID\_Stato)

REFERENCES State (ID\_Stato)

);

CREATE TABLE Sales (

ID\_Sales INT,

ID\_Regione INT,

ID\_Prodotto INT,

Data\_ordine TIMESTAMP NOT NULL,

Quantita INT NOT NULL,

Fatturato INT NOT NULL,

CONSTRAINT PK\_Sales PRIMARY KEY (ID\_Sales),

CONSTRAINT FK2\_State FOREIGN KEY (ID\_Regione)

REFERENCES Region (ID\_Regione),

CONSTRAINT FK2\_Product FOREIGN KEY (ID\_Prodotto)

REFERENCES Product (ID\_Prodotto)

);

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

INSERT INTO category (ID\_Categoria,Desc\_Categoria) VALUES (1,'Bambole');

INSERT INTO category (ID\_Categoria,Desc\_Categoria) VALUES (2,'Macchinine');

INSERT INTO category (ID\_Categoria,Desc\_Categoria) VALUES (3,'Giochi da tavolo');

INSERT INTO category (ID\_Categoria,Desc\_Categoria) VALUES (4,'Costruzioni');

INSERT INTO Product (ID\_Prodotto,ID\_Categoria,Nome\_Prodotto,Prezzo\_unitario)VALUES(1,1,'Barbie', 30);

INSERT INTO Product (ID\_Prodotto,ID\_Categoria,Nome\_Prodotto,Prezzo\_unitario)VALUES(2,1,'Bratz', 45);

INSERT INTO Product (ID\_Prodotto,ID\_Categoria,Nome\_Prodotto,Prezzo\_unitario)VALUES(3,1,'Cicciobello', 60);

INSERT INTO Product (ID\_Prodotto,ID\_Categoria,Nome\_Prodotto,Prezzo\_unitario)VALUES(4,2,'Hotweels', 10);

INSERT INTO Product (ID\_Prodotto,ID\_Categoria,Nome\_Prodotto,Prezzo\_unitario)VALUES(5,2,'Ferrari', 15);

INSERT INTO Product (ID\_Prodotto,ID\_Categoria,Nome\_Prodotto,Prezzo\_unitario)VALUES(6,3,'Monopoly', 20);

INSERT INTO Product (ID\_Prodotto,ID\_Categoria,Nome\_Prodotto,Prezzo\_unitario)VALUES(7,3,'Risiko', 8);

INSERT INTO Product (ID\_Prodotto,ID\_Categoria,Nome\_Prodotto,Prezzo\_unitario)VALUES(8,3,'Gioco oca', 7);

INSERT INTO Product (ID\_Prodotto,ID\_Categoria,Nome\_Prodotto,Prezzo\_unitario)VALUES(9,4,'Lego', 25);

INSERT INTO Product (ID\_Prodotto,ID\_Categoria,Nome\_Prodotto,Prezzo\_unitario)VALUES(10,4,'Geomag',20);

INSERT INTO State (ID\_Stato,Nome\_stato) VALUES(1,'Europa');

INSERT INTO State (ID\_Stato,Nome\_stato) VALUES(2,'Africa');

INSERT INTO State (ID\_Stato,Nome\_stato) VALUES(3,'America');

INSERT INTO State (ID\_Stato,Nome\_stato) VALUES(4,'Australia');

INSERT INTO region (ID\_Regione,ID\_Stato,Nome\_regione) VALUES(1,1,'Italia');

INSERT INTO region (ID\_Regione,ID\_Stato,Nome\_regione) VALUES(2,1,'Germania');

INSERT INTO region (ID\_Regione,ID\_Stato,Nome\_regione) VALUES(3,2,'Marocco');

INSERT INTO region (ID\_Regione,ID\_Stato,Nome\_regione) VALUES(4,2,'Somalia');

INSERT INTO region (ID\_Regione,ID\_Stato,Nome\_regione) VALUES(5,2,'Madagascar');

INSERT INTO region (ID\_Regione,ID\_Stato,Nome\_regione) VALUES(6,3,'USA');

INSERT INTO region (ID\_Regione,ID\_Stato,Nome\_regione) VALUES(7,3,'Messico');

INSERT INTO region (ID\_Regione,ID\_Stato,Nome\_regione) VALUES(8,3,'Venezuela');

INSERT INTO region (ID\_Regione,ID\_Stato,Nome\_regione) VALUES(9,4,'Nuova Zelanda');

INSERT INTO region (ID\_Regione,ID\_Stato,Nome\_regione) VALUES(10,4,'Nuova Guinea');

INSERT INTO Sales(ID\_Sales, ID\_Regione, ID\_Prodotto, Data\_ordine, Quantita, Fatturato) VALUES(1, 1,1,'2023-11-01',2,60);

INSERT INTO Sales(ID\_Sales, ID\_Regione, ID\_Prodotto, Data\_ordine, Quantita, Fatturato) VALUES(2, 1,2,'2023-12-01',1,45);

INSERT INTO Sales(ID\_Sales, ID\_Regione, ID\_Prodotto, Data\_ordine, Quantita, Fatturato) VALUES(3, 2,3,'2023-01-01',3,180);

INSERT INTO Sales(ID\_Sales, ID\_Regione, ID\_Prodotto, Data\_ordine, Quantita, Fatturato) VALUES(4, 2,4,'2023-01-12',2,20);

INSERT INTO Sales(ID\_Sales, ID\_Regione, ID\_Prodotto, Data\_ordine, Quantita, Fatturato) VALUES(5, 3,5,'2019-03-15',1,15);

INSERT INTO Sales(ID\_Sales, ID\_Regione, ID\_Prodotto, Data\_ordine, Quantita, Fatturato) VALUES(6, 3,6,'2020-07-22',1,20);

INSERT INTO Sales(ID\_Sales, ID\_Regione, ID\_Prodotto, Data\_ordine, Quantita, Fatturato) VALUES(7, 4,7,'2021-11-05',5,40);

INSERT INTO Sales(ID\_Sales, ID\_Regione, ID\_Prodotto, Data\_ordine, Quantita, Fatturato) VALUES(8, 4,8,'2018-06-12',2,14);

INSERT INTO Sales(ID\_Sales, ID\_Regione, ID\_Prodotto, Data\_ordine, Quantita, Fatturato) VALUES(9, 5,9,'2022-09-30',3,75);

INSERT INTO Sales(ID\_Sales, ID\_Regione, ID\_Prodotto, Data\_ordine, Quantita, Fatturato) VALUES(10,5,1,'2023-04-18',2,60);

INSERT INTO Sales(ID\_Sales, ID\_Regione, ID\_Prodotto, Data\_ordine, Quantita, Fatturato) VALUES(11,6,2,'2017-08-07',1,45);

INSERT INTO Sales(ID\_Sales, ID\_Regione, ID\_Prodotto, Data\_ordine, Quantita, Fatturato) VALUES(12,6,3,'2020-01-25',4,240);

INSERT INTO Sales(ID\_Sales, ID\_Regione, ID\_Prodotto, Data\_ordine, Quantita, Fatturato) VALUES(13,7,4,'2021-10-14',1,10);

INSERT INTO Sales(ID\_Sales, ID\_Regione, ID\_Prodotto, Data\_ordine, Quantita, Fatturato) VALUES(14,7,5,'2018-05-09',1,15);

INSERT INTO Sales(ID\_Sales, ID\_Regione, ID\_Prodotto, Data\_ordine, Quantita, Fatturato) VALUES(15,8,6,'2022-02-08',2,40);

INSERT INTO Sales(ID\_Sales, ID\_Regione, ID\_Prodotto, Data\_ordine, Quantita, Fatturato) VALUES(16,8,7,'2017-09-16',5,40);

INSERT INTO Sales(ID\_Sales, ID\_Regione, ID\_Prodotto, Data\_ordine, Quantita, Fatturato) VALUES(17,9,8,'2019-12-03',3,21);

INSERT INTO Sales(ID\_Sales, ID\_Regione, ID\_Prodotto, Data\_ordine, Quantita, Fatturato) VALUES(18,9,9,'2020-07-28',2,50);

INSERT INTO Sales(ID\_Sales, ID\_Regione, ID\_Prodotto, Data\_ordine, Quantita, Fatturato) VALUES(19,10,5,'2021-05-11',1,15);

INSERT INTO Sales(ID\_Sales, ID\_Regione, ID\_Prodotto, Data\_ordine, Quantita, Fatturato) VALUES(20,10,7,'2018-11-20',1,8);

INSERT INTO Sales(ID\_Sales, ID\_Regione, ID\_Prodotto, Data\_ordine, Quantita, Fatturato) VALUES(21,7,2,'2023-10-11',2,90);

INSERT INTO Sales(ID\_Sales, ID\_Regione, ID\_Prodotto, Data\_ordine, Quantita, Fatturato) VALUES(22,1,8,'2023-10-20',58, 406);

**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).

SELECT

ID\_Prodotto, COUNT(\*) AS Conteggio

FROM

Product

GROUP BY ID\_Prodotto

HAVING Conteggio > 1;

SELECT

ID\_Categoria, COUNT(\*) AS Conteggio

FROM

Category

GROUP BY ID\_Categoria

HAVING Conteggio > 1;

SELECT

ID\_Sales, COUNT(\*) AS Conteggio

FROM

Sales

GROUP BY ID\_Sales

HAVING Conteggio > 1;

SELECT

ID\_Stato, COUNT(\*) AS Conteggio

FROM

State

GROUP BY ID\_Stato

HAVING Conteggio > 1;

SELECT

ID\_Regione, COUNT(\*) AS Conteggio

FROM

Region

GROUP BY ID\_Regione

HAVING Conteggio > 1;

1. 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

S.ID\_Sales,

S.Data\_Ordine,

P.Nome\_Prodotto,

C.Desc\_categoria,

St.Nome\_Stato,

R.Nome\_regione,

CASE

WHEN date(S.DATA\_ORDINE) > (curdate() -180) THEN true

ELSE false

END AS VALORE

FROM

Sales AS S

INNER JOIN

Product AS P ON S.ID\_Prodotto = P.ID\_Prodotto

INNER JOIN

Category AS C ON P.ID\_Categoria = C.ID\_Categoria

INNER JOIN

Region AS R ON S.ID\_Regione = R.ID\_Regione

INNER JOIN

State AS St ON R.ID\_Stato = St.ID\_Stato;

1. 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

ID\_Prodotto, SUM(quantita) AS Tot\_qtyvenduta

FROM

Sales

GROUP BY ID\_Prodotto

HAVING Tot\_qtyvenduta > (SELECT

AVG(quantita) AS Media\_qtyvenduta

FROM

Sales

WHERE

YEAR(Data\_ordine) = (SELECT

MAX(YEAR(Data\_ordine))

FROM

sales));

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

SELECT

P.Nome\_Prodotto,

SUM(S.Fatturato) AS Totale\_Fatturato,

YEAR(S.Data\_ordine) AS Anno

FROM

Sales AS S

INNER JOIN

Product AS P ON S.ID\_Prodotto = P.ID\_Prodotto

GROUP BY P.Nome\_Prodotto , Anno;

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

SELECT

St.Nome\_Stato,

SUM(S.Fatturato) AS Tot\_Fatturato,

YEAR(S.Data\_Ordine) AS Anno

FROM

Sales AS S

INNER JOIN

Region AS R ON S.ID\_Regione = R.ID\_Regione

INNER JOIN

State AS St ON R.ID\_Stato = St.ID\_Stato

GROUP BY St.Nome\_Stato, Anno

ORDER BY Anno DESC , Tot\_Fatturato DESC;

1. Rispondere alla seguente domanda: qual è la categoria di articoli maggiormente richiesta dal mercato?

SELECT

C.desc\_categoria, SUM(S.Quantita) AS Conteggio\_vendite

FROM

Sales AS S

INNER JOIN

Product AS P ON S.ID\_Prodotto = P.ID\_Prodotto

INNER JOIN

Category AS C ON P.ID\_Categoria = C.ID\_Categoria

GROUP BY C.desc\_categoria

ORDER BY Conteggio\_vendite DESC;

-- La categoria maggiormente richiesta dal mercato è Giochi a tavolo --

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

**APPROCCIO 1: JOIN**

SELECT

P.Nome\_Prodotto

FROM

Product AS P

LEFT JOIN

Sales AS S ON P.ID\_Prodotto = S.ID\_Prodotto

WHERE

S.ID\_Prodotto IS NULL;

**APPROCCIO 2: SUBQUERY**

SELECT

Nome\_Prodotto

FROM

Product

WHERE

ID\_Prodotto NOT IN (SELECT

ID\_Prodotto

FROM

Sales);

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

CREATE VIEW Vista\_Den AS

(SELECT

P.ID\_Prodotto, P.Nome\_Prodotto, C.Desc\_Categoria

FROM

Product AS P

INNER JOIN

Category AS C ON P.ID\_Categoria = C.ID\_Categoria);

1. Creare una vista per le informazioni geografiche

CREATE VIEW Vista\_geografica AS

(SELECT

R.Nome\_regione, R.ID\_Regione, St.Nome\_stato, St.ID\_Stato

FROM

Region AS R

INNER JOIN

State AS St ON R.ID\_Stato = St.ID\_Stato);