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?

RISPOSTE:

1)Un DB è un insieme di archive informatizzati connessi tra loro, che rende possible la consultazione e l'aggiornamento in tempo reale delle informazioni.

2) E' un sistema software progettato per consentire la creazione, la manipolazione e l'interrogazione di database.

3) FROM, indica la tabella dalle quale si interrogano i record; WHERE, filtra in base alle condizioni di ricercar le righe restituite da from; GROUP BY, raggruppa i record restituiti precedentemente per ogni combinazione univocal dei campi delle 'liste' indicate; HAVING, filtra in case alle condizioni di ricercar I gruppi restituiti da group by; SELECT, indica I campi da restituire nel result set; ORDER BY, ordinamenti.

4) Il GROUP BY consente di creare dei gruppi. E' spesso utilizzato in combinazione alle funzioni di aggregazione (sum, avg, count,..). Immaginando di avere una tabella 'prodotti' avente campi 'nome prodotto', 'categoria' e 'prezzo', la query "SELECT categoria, avg(prezzo) FROM prodotti GROUP BY categoria;" restituirà un result set che mostra il prezzo medio per categoria".

5) OLTP garantisce la consistenza, l'integrità e la sicurezza delle transazioni. L'OLAP è l'approccio metodologico e riguardante l'utilizzo software afferente all'analisi complessi di big data a support della BI.

6) Sia una join che una subquery possono essere utilizzate per combinare dati da diverse tabelle e ottenere risultati simili. Una join mi permette di 'unire' due tabelle basato su una condizione di corrispondenza tra tabelle mentre le subquery sono query innestate all'interno di altre e possono essere utilizzate per filtrare i dati nella query principale o per ottenere valori che possono essere usati nella query principale.

7) Sono due tipologie di comandi SQL: Data Manipulation Language, si usano per alterare, modificare I dati (ad es. Insert, update, delete) e Data Defination Language, per strutturali (ad es. Create, drop, alter).

8) Si può usare la funzione YEAR() che prende un valore di tipo date, datetime e restituisce solo l'anno come numero. Immaginando di avere una tabella 'ordini' con un campo 'dataordine', utilizzando la query 'SELECT dataordine, YEAR(DATAORDINE) as anno FROM ordini, otterò un result set in cui nel campo 'anno' ho il rispettivo anno dell'ordine. Potrei anche estrarre una data come stringa se nella select riportassi year(istanza_del_campo).

9) AND richiede che tutte le condizioni siano vere perché sia vera un'espression, a OR ne basta una perché sia vera.

10) Sì. In questo caso parliamo di una subquery scalare autonoma.

11) OR, oltre a ciò detto prima, può unire condizioni o confronti diversi mentre IN verifica se un valore è in un insieme di valori specifici.

12) Sì. Se un range specificato fosse tra 10 e 20, qualora quelle istanze rientrassero nelle condizioni specificate, verrebbero incluse.

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

2) 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

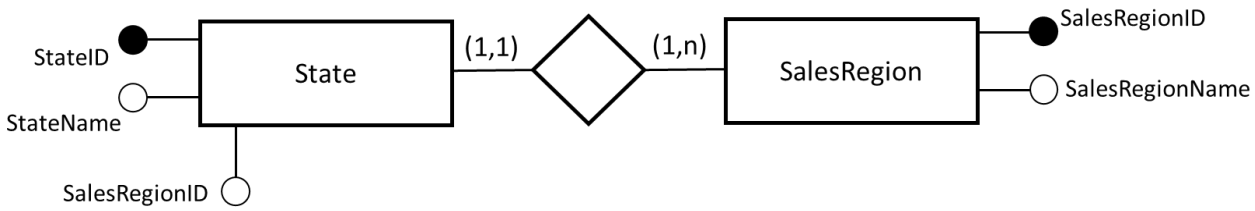
3) Le entità Product e Region presentano delle gerarchie:
    - L'entità prodotto contiene, oltre alle informazioni del singolo prodotto, anche la descrizione della categoria di appartenenza. L'entità prodotto contiene quindi una gerarchia: un prodotto puo' appartenere ad una sola categoria mentre la 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.

Sales		
OrderID	*OrderDate*	*StateID*
1	01/01/2023	1
2	01/01/2023	2

n

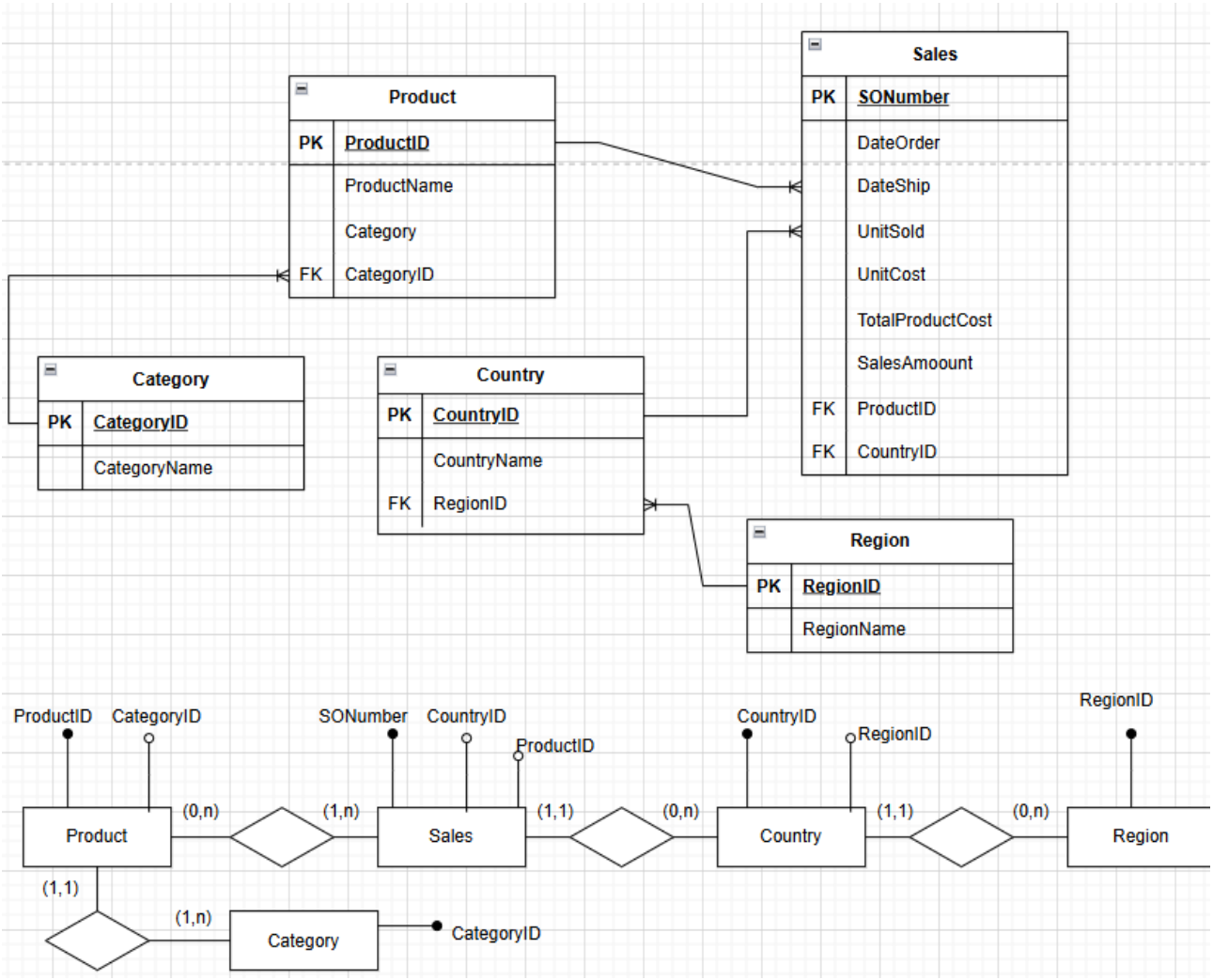
1

State		
StateID	*StataName*	*SalesRegionID*
1	Italy	1
2	Germany	2

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

TASK 1

Nell'immagine allegata qui sotto sono presenti sia il modello logico, che include tutti i campi (attributi), le entità e le loro relazioni, sia il diagramma di cardinalità, in cui sono rappresentate le entità con le rispettive chiavi primarie e chiavi esterne, insieme alle cardinalità delle relazioni che ho ritenuto corrette.



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 database ToysGroup; #creo il db
create table Category (
CategoryID int auto_increment primary key,
CategoryName varchar(50) not null
);
create table Region (
RegionID int auto_increment primary key,
RegionName varchar(50) not null
);
create table Product (
ProductID INT AUTO_INCREMENT primary key,
ProductName varchar(50) not null,
CategoryID INT not null,
FOREIGN KEY (CategoryID) references Category(CategoryID)
);
create table Country (
CountryID int auto_increment primary key,
CountryName varchar(50) not null,
RegionID int not null,
foreign key (RegionID) REFERENCES Region(RegionID)
);
```

```
	create table Sales (
	SONumber int auto_increment primary key,
	DateOrder date not null,
	DateShip date not null,
	UnitSold decimal(10,2) not null,
	UnitCost decimal(10,2) not null,
	TotalProductCost decimal(10,2) not null,
	SalesAmount decimal(10,2) not null,
	ProductID int not null,
	CountryID INT not null,
	foreign key (ProductID) REFERENCES Product(ProductID),
	Foreign key (CountryID) references Country(CountryID)
);
```

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

```
	INSERT INTO Category (CategoryName) VALUES
	('Costruzioni'),
	('Bambole'),
	('Veicoli'),
	('Puzzle'),
	('Educativi')
	;
	INSERT INTO Region (RegionName) VALUES
	('NorthEurope'),
	('SouthEurope'),
	('WestEurope')
	;
	INSERT INTO Country (CountryName, RegionID) VALUES
	('Germany', 3),
	('France', 3),
	('Italy', 2),
	('Spain', 2),
	('Norway', 1),
	('Sweden', 1)
	;
	INSERT INTO Product (ProductName, CategoryID) VALUES
	('Lego Classic 11005', 1),
	('Barbie Fashionista', 2),
	('Hot Wheels Mega Garage', 3),
	('Puzzle Mappamondo 1000pz', 4),
	('Set Didattico STEM Junior', 5),
	('Lego Technic Bugatti', 1),
	('Bambola Frozen Elsa', 2),
	('Treno Elettrico Express', 3)
	;
	INSERT INTO Sales (DateOrder, DateShip, UnitSold, UnitCost, TotalProductCost,
	SalesAmount, ProductID, CountryID) VALUES
	('2024-01-15', '2024-01-17', 10, 15.00, 150.00, 250.00, 1, 3),
	('2024-02-10', '2024-02-12', 5, 22.00, 110.00, 200.00, 2, 3),
	('2024-03-01', '2024-03-03', 12, 12.00, 144.00, 240.00, 3, 4),
	('2024-01-25', '2024-01-27', 3, 18.00, 54.00, 90.00, 4, 1),
	('2023-12-15', '2023-12-17', 6, 20.00, 120.00, 180.00, 5, 2),
	('2024-04-10', '2024-04-12', 8, 35.00, 280.00, 480.00, 6, 5),
	('2024-05-01', '2024-05-02', 4, 25.00, 100.00, 180.00, 7, 6),
	('2024-05-20', '2024-05-22', 7, 30.00, 210.00, 350.00, 8, 1)
	;
```

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 ProductID, COUNT(*) FROM Product GROUP BY ProductID HAVING COUNT(*) > 1;
SELECT CategoryID, COUNT(*) FROM Category GROUP BY CategoryID HAVING COUNT(*) > 1;
SELECT Sonumber, COUNT(*) FROM Sales GROUP BY sonumber HAVING COUNT(*) > 1;
SELECT RegionID, COUNT(*) FROM REGION GROUP BY REGIONID HAVING COUNT(*) > 1;
SELECT cOUNTRYID, COUNT(*) FROM COUNTRY GROUP BY Countryid HAVING COUNT(*) > 1;
#se i resultset sono vuoti, vuol dire che le chiavi sono univoche
```

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 s.sonumber as Codicedocumento, s.dateorder, p.productname, cat.categoryname,
c.countryname, r.regionname, DATEDIFF(S.dateship, S.DateOrder) > 180 AS TorF
from sales s
join product p on s.productid = p.productid
join country c on s.countryid = c.countryid
join region r on c.regionid = r.regionid
join category cat on p.categoryid = cat.categoryid;
#con datediff >180 stabilisco se saranno veri(1), alrimenti falsi(0)

!!!Devo apportare delle modifiche poiché per i dati che avevo inserito, nessuno era vero
(nesssuna diff era maggiore di 180 giorni):

set autocommit = 0;    #disattivo l'autocommit di mysql
start transaction;
update sales
set dateship = '2024-08-19'
where sonumber = 2;
update sales
set dateship = '2024-06-19'
where sonumber = 3;
update sales
set dateship = '2024-08-19'
where sonumber = 1;
select * from sales;
commit;       #Ho modificato delle istanze di dateship
```

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(UnitSold) AS Totalepezzivenduto
FROM SALES
WHERE YEAR(Dateorder) = (SELECT MAX(YEAR(DateOrder)) FROM Sales)
GROUP BY ProductID
HAVING SUM(UnitSold) > (SELECT AVG(SommaVendite)
FROM (SELECT SUM(UNITSOLD) AS SommaVendite
FROM Sales
WHERE YEAR(DateOrder) = (
SELECT MAX(YEAR(DateOrder)) FROM Sales)
GROUP BY ProductID) AS SubVendite);
#la prima subquery mi permette di utilizzare l'ultimo anno censito come filtro. La doppia subquery nell'having calcola la media delle somme annuali e le confronta con ogni somma per prodotto
```

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

```
Select p.productname, sum(s.Salesamount) as fatturato, year(dateorder) as Anno
from sales s
join product p on s.productid = p.productid
where dateorder is not null
group by p.productname, year(dateorder)
order by anno desc;
```

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

```
Select year(DATEORDER) as Anno, sum(SALESAMOUNT) as Fatturato, countryname
from Sales s
join COUNTRY C ON S.COUNTRYID = C.COUNTRYID
group by countryname, year(dateorder)
ORDER BY year(dateorder), fatturato DESC;
```

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

```
select cat.categoryname, sum(s.unitsold) as Pezzivenduti
from sales s
join product p on s.productid = p.productid
join category cat on p.categoryid = cat.categoryid
group by cat.categoryname
order by pezzivenduti desc
limit 1;
 #ordinando in desc e limitando il result set alla prima riga, ottengo la categoria con più unita vendute
```

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

```
SELECT p.ProductID, p.ProductName
FROM Product p
LEFT JOIN Sales s ON p.ProductID = s.ProductID
```

```
WHERE s.ProductID IS NULL;
#con left join escludo dal result set i record di sales in cui non appaiono i produtti non
venduti (nella tabella sales ci sono solo prod. venduti)

SELECT productID, productname
FROM PRODUCT
WHERE ProductID NOT IN (SELECT DISTINCT ProductID FROM Sales);
#con questa subquery ottengo un result set con prodotti non presenti nelle vendite

#Ho dovuto inserire due prodotti per fare in modo che i due result set delle query
precedenti non fossero vuoti (avevo immesso dati che mi davano solo prodotti venduti):

INSERT INTO Product (ProductName, CategoryID)
VALUES
  ('Puzzle Torre Eiffel 3000pz', 4),
  ('Camion dei Pompieri PlaySet', 3);
```

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 Versionedenormalizzata as (select productid as CodiceProdotto, productname as
NomeProdotto, (select categoryname from category cat where p.categoryid = cat.categoryid) as
NomeCategoria
 from product p);
 select * from versionedenormalizzata;     #view creata utilizzando una subquery
```

9) Creare una vista per le informazioni geografiche

```
create view InfoGeografiche as (select C.CountryID AS CodiceStato, c.Countryname as Stato,
r.Regionname as Regione
 from country c
 join region r on c.regionid = r.regionid);
 select * from infogeografiche;             #view creata utilizzando inner join
```

Allego con questo documento lo script con lo svolgimento delle Task 2, 3 e 4.