Struttura tabelle

```
// Clienti

CREATE TABLE Clienti{

IDCliente INT PRIMARY KEY AUTOINCREMENT,

// PRIMARY KEY = chiave primaria = identifica una tabella

// AUTOINCREMENT = numero progressivo = ad ogni nuovo inserimento aumenta

Nome VARCHAR(50) NOT NULL,
Cognome VARCHAR(50) NOT NULL,
Citta VARCHAR(50) NOT NULL,
Salario INT,
Eta INT

// VARCHAR = Insieme di caratteri
// Es. Nome ha 50 caratteri
// NOT NULL = non nullo = non vuoto

);
```

Entità e collegamenti

Ecco un indice Markdown sintetizzato del contenuto:

Database Relazionale: Proprietari e Auto

```
CREATE TABLE Proprietario(
    Idpro INT AUTO_INCREMENT PRIMARY KEY,
    Nome VARCHAR(50) NOT NULL,
    Cognome VARCHAR(50) NOT NULL,
    DataN DATE NOT NULL,
    Indirizzo VARCHAR(50) NOT NULL,
```

```
    Telefono VARCHAR(50)
);
```

2.2 Tabella Auto

```
CREATE TABLE Auto(
    Targa VARCHAR(7) PRIMARY KEY NOT NULL,
    Marca VARCHAR(50) NOT NULL,
    Modello VARCHAR(50) NOT NULL,
    Colore VARCHAR(50) NOT NULL,
    Cilindrata INT NOT NULL,
    Potenza INT NOT NULL,
    Idpro INT,
    FOREIGN KEY (Idpro) REFERENCES Proprietario (Idpro)
);
```

Relazioni tra Entità

- Tipo: 1 a molti (un proprietario può possedere più auto)
- Collegamento: dalla chiave primaria in PROPRIETARIO verso la chiave esterna in AUTO
- Integrità referenziale: applicata tramite FOREIGN KEY
- Nota: PRIMARY KEY e FOREIGN KEY devono essere dello stesso tipo (INT in questo caso)

Selezioni e proiezioni

La struttura di una interrogazione con **proiezione** è la seguente:

```
SELECT         campi di output (colonne)
FROM           entità di provenienza (la tabella da cui provengono i dati)
WHERE         condizione di estrazione (è come fare un "if")
```

	Clienti				
IdCliente	Cognome	Nome	Citta	Salario	Eta
1	Bianchi	Mario	Rimini	1000	20
2	Bianchi	Ettore	Milano	2500	15
3	Casadei	Mario	Rimini	3000	35
4	Rossi	Mario	Bologna	1500	50
5	Rossi	Fabio	Firenze	8000	40
6	Bianchi	Ettore	Rimini	4500	25
7	Neri	Fabio	Arezzo	3500	35

Esercizi:

1) Scrivere la query che restituisce nome e cognome di chi guadagna più di 3000;

```
SELECT Nome, Cognome
FROM Clienti
WHERE Salario > 3000;
```

2. Scrivere la query che restituisce cognome e nome dei clienti che abitano a Rimini;

```
SELECT Nome, Cognome
FROM Clienti
WHERE Citta = "Rimini";
```

Operatori logici

- AND prodotto logico
- OR somma logica
- NOT negazione

1) Scrivere la query che restituisce cognome e nome dei clienti che abitano a Rimini E guadagnano più di 3000;

```
SELECT Nome, Cognome
FROM Clienti
WHERE Citta = "Rimini" AND Salario > 3000;
```

1) Scrivere la query che restituisce cognome, nome e salario dei clienti che hanno età compresa fra 20 e 35 anni (estremi compresi);

```
SELECT Cognome, Nome, Salario
FROM Clienti
```

```
WHERE Eta >= 20 AND Eta <= 35;

// Alternativa = usare BETWEEN (in inglese "tra")

SELECT Cognome, Nome, Salario
FROM Clienti
WHERE Eta BETWEEN 20 AND 35;

// Nota: BETWEEN di solito si usa per i valori numerici o per le date

// DATE: 2024-10-15

// Es -> Query per trovare i clienti che hanno guadagnato più di 3000 tra il
15 e il 30 ottobre

// * = asterisco = prendi "tutto"

SELECT *
FROM Clienti
WHERE Data_Guadagno BETWEEN "2024-10-15" AND "2024-10-31";

// CREATE TABLE ...
```

1) Scrivere la query che restituisce cognome, nome e salario dei clienti che risiedono a Rimini e hanno meno di 20 anni O più di 30.

```
SELECT    Cognome, Nome, Salario
FROM       Cliente
WHERE    Citta = "Rimini" AND (Eta<20 OR Eta>30);
```

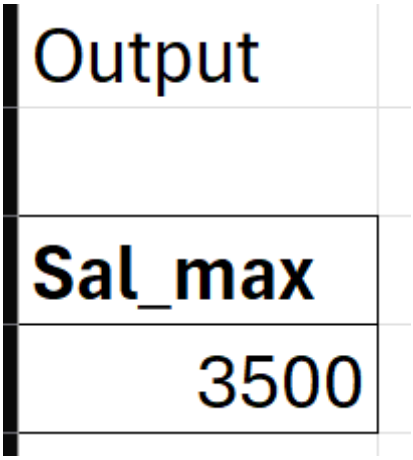
Funzioni di campo calcolato

- MAX = trova il massimo di una colonna
- MIN = trova il minimo di una colonna
- AVG = (average=medio) calcola la media
- SUM = somma
- COUNT = conta il numero di colonne che soddisfa una certa condizione

NOTA: tutte queste funzioni agiscono **sulle colonne**. Quindi, le mettiamo nel SELECT

1. Scrivere la query che restituisce il salario massimo

```sql
SELECT MAX(Salario) AS Salario_Massimo
// AS = dà un soprannome alla colonna
FROM Cliente
```



2. Scrivere la query che restituisce il salario minimo dei clienti residenti a Rimini con età compresa tra 25 e 40 anni (estremi inclusi);

```sql
SELECT MIN(Salario) as Salario_minimo
FROM Cliente
WHERE Età BETWEEN 25 AND 40;
```

4. Scrivere la query che restituisce il numero di clienti che hanno età minore di 25 anni o maggiore di 35.

```sql
SELECT COUNT(*)
FROM Cliente;
WHERE (Età < 25 OR Età > 35);
```

6. Scrivere la query che restituisce il Nome e il Cognome del/dei clienti con salario MAX.

```sql
SELECT Nome, Cognome, Salario FROM Cliente GROUP BY Nome, Cognome, Salario
HAVING Salario = (SELECT MAX(Salario) FROM Cliente);

// Se non usiamo GROUP BY non facciamo vedere tutti i dati

- GROUP BY raggruppa le righe di un risultato che hanno gli stessi valori
nelle colonne specificate.
- Viene usato con funzioni di aggregazione (come COUNT, MAX, MIN, SUM, AVG)
```

```
- HAVING è usato per filtrare i risultati di un GROUP BY.
- Funziona come WHERE, ma si applica ai gruppi dopo che sono stati formati,
permettendo di filtrare in base ai risultati di funzioni di aggregazione.

// Sottoquery = uso una query dipendente dal campo - soluzione di Campagnaro

SELECT Nome, Cognome
FROM    Cliente
WHERE Salario = (SELECT MAX(Salario) FROM Cliente);
```

8. Visualizzare Nome, Cognome ed età dei clienti con salario maggiore della media.

```
SELECT Nome, Cognome, Età
FROM Cliente
WHERE Salario > (SELECT AVG(Salario) AS Salario_medio FROM Cliente)
```

10. Visualizzare Nome e Cognome delle persone con età maggiore alla media.

```
SELECT Nome, Cognome
FROM Cliente
WHERE Età > (SELECT AVG(Età) AS Età_media FROM Cliente)
```