

## CREATE

- **Mit csinál?** Új adatbázis vagy tábla létrehozása.
- **Példa:**

-- Adatbázis létrehozása

```
CREATE DATABASE Iskola;
```

-- Tábla létrehozása

```
CREATE TABLE Diakok (  
    DiakID INT PRIMARY KEY,  
    Nev VARCHAR(50),  
    Osztaly VARCHAR(10)  
);
```

## DROP

- **Mit csinál?** Meglévő adatbázis vagy tábla törlése.
- **Példa:**

DROP TABLE Diakok; -- Tábla törlése

DROP DATABASE Iskola; -- Adatbázis törlése

## INSERT INTO

- **Mit csinál?** Új rekordok beszúrása egy táblába.
- **Példa:**

```
INSERT INTO Diakok (DiakID, Nev, Osztaly)  
VALUES (1, 'Kovács Anna', '10.A');
```

## SELECT

- **Mit csinál?** Adatok lekérdezése egy táblából.
- **Példa:**

SELECT \* FROM Diakok; -- Összes adat

SELECT Nev FROM Diakok WHERE Osztaly = '10.A'; -- Csak a 10.A diákjai

---

## 2. Táblák Összekapcsolása: JOIN

### INNER JOIN

- **Mit csinál?** Csak azokat a rekordokat adja vissza, ahol mindkét táblában van egyezés.
- **Példa (Diakok + Jegyek tábla):**

```
SELECT Diakok.Nev, Jegyek.Jegy
```

```
FROM Diakok
```

```
INNER JOIN Jegyek ON Diakok.DiakID = Jegyek.DiakID;
```

### LEFT JOIN

- **Mit csinál?** A bal oldali tábla összes rekordját mutatja, a jobb oldali tábla mezőit csak ha van egyezés.
- **Példa:**

```
SELECT Diakok.Nev, Jegyek.Jegy
```

```
FROM Diakok
```

```
LEFT JOIN Jegyek ON Diakok.DiakID = Jegyek.DiakID;
```

---

## 1. Lekérdezések Finomhangolása

### WHERE (Szűrés)

- **Mit csinál?** Szűrés adott feltétel alapján.
- **Példa:**

```
SELECT * FROM Diakok WHERE Osztaly = '10.B';
```

### ORDER BY (Rendezés)

- **Mit csinál?** Adatok rendezése növekvő (ASC) vagy csökkenő (DESC) sorrendben.
- **Példa:**

```
SELECT Nev, SzulDatum FROM Diakok ORDER BY SzulDatum DESC; -- Legfiatalabb elől
```

## GROUP BY (Csoportosítás)

- **Mit csinál?** Adatok csoportosítása mező alapján. Gyakran aggregáló függvényekkel használják.
- **Példa:**

```
SELECT Osztaly, COUNT(*) AS DiakokSzama
```

```
FROM Diakok
```

```
GROUP BY Osztaly; -- Diákok száma osztályonként
```

---

## 4. Aggregáló Függvények

### SUM

- **Mit csinál?** Összeg kiszámítása.
- **Példa:**

```
SELECT SUM(Jegy) AS OsszPontszam FROM Jegyek WHERE DiakID = 1;
```

### COUNT

- **Mit csinál?** Rekordok számolása.
- **Példa:**

```
SELECT COUNT(*) AS OsszesDiak FROM Diakok;
```

### AVG

- **Mit csinál?** Átlagérték számítása.
- **Példa:**

```
SELECT AVG(Jegy) AS AtlagJegy FROM Jegyek;
```

### MIN/MAX

- **Mit csinál?** Legkisebb/legnagyobb érték megtalálása.
- **Példa:**

```
SELECT MIN(Jegy) AS LegrosszabbJegy FROM Jegyek;
```

```
SELECT MAX(Jegy) AS LegjobbJegy FROM Jegyek;
```

## 5. Szöveges Függvények

### LEFT/RIGHT

- **Mit csinál?** Szöveg bal/jobbr oldalának kivágása.
- **Példa:**

```
SELECT LEFT(Nev, 3) AS RovidNev FROM Diakok; -- Első 3 karakter
```

```
SELECT RIGHT(Nev, 4) FROM Diakok; -- Utolsó 4 karakter
```

### CONCAT

- **Mit csinál?** Szövegek összefűzése.
- **Példa:**

```
SELECT CONCAT(Nev, ' (' , Osztaly, ')') AS TeljesNev FROM Diakok;
```

### LENGTH

- **Mit csinál?** Szöveg hosszának mérése.
- **Példa:**

```
SELECT Nev, LENGTH(Nev) AS Hossz FROM Diakok;
```

---

## 6. Gyakori Hibák & Típek

### 1. Hibás szintaxis:

- **Rossz:** SELECT \* Diakok → **Jó:** SELECT \* FROM Diakok.

### 2. GROUP BY elfelejtése aggregáló függvényeknél:

-- Helytelen:

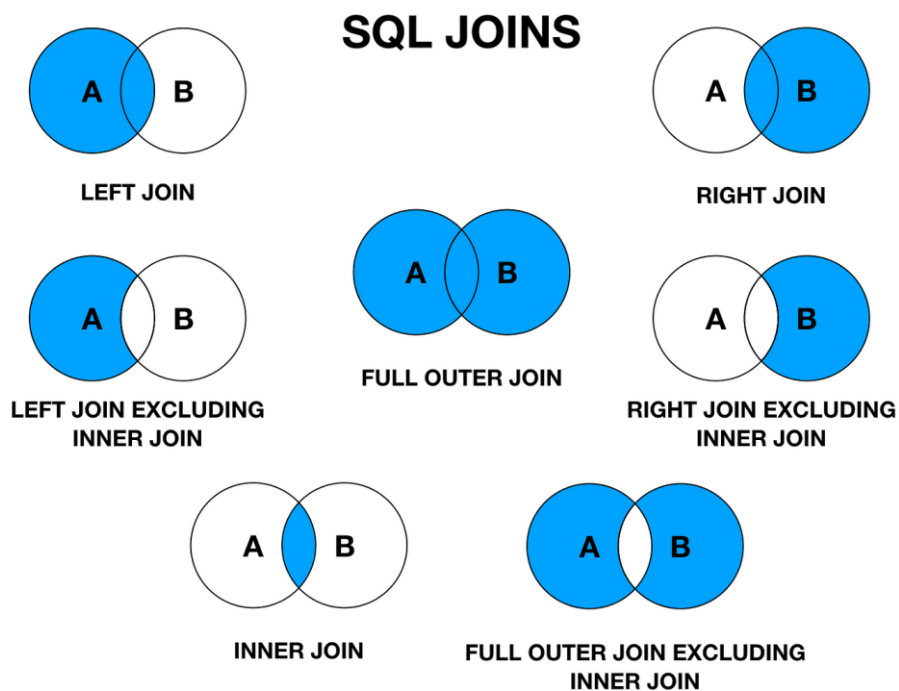
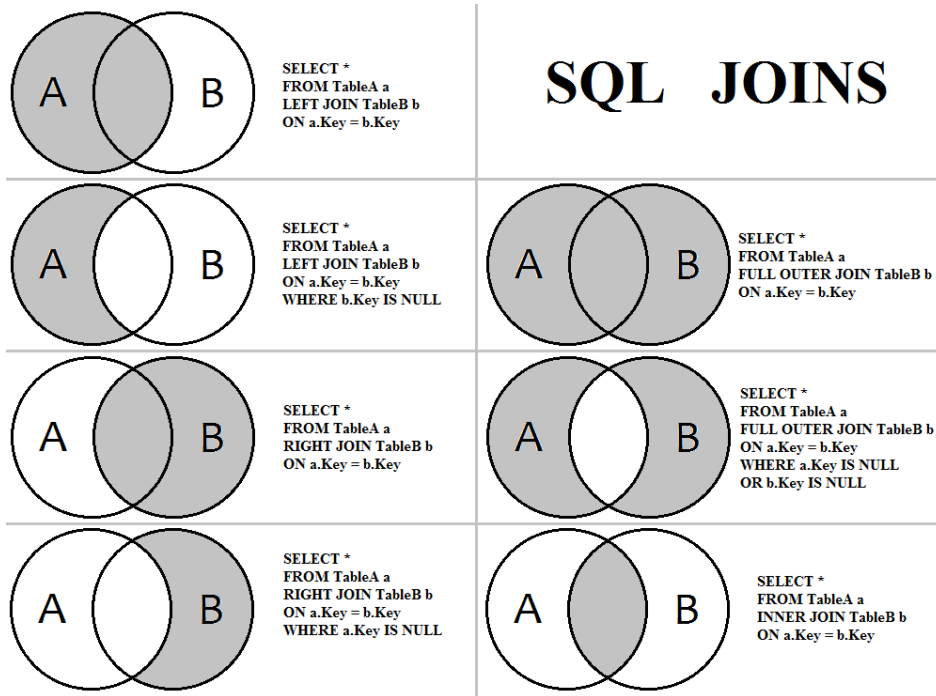
```
SELECT Osztaly, COUNT(*) FROM Diakok;
```

-- Helyes:

```
SELECT Osztaly, COUNT(*) FROM Diakok GROUP BY Osztaly;
```










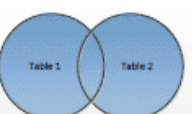




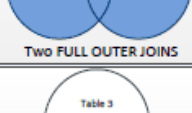
### 3. WHERE vs. HAVING:

- **WHERE:** Szűrés **csoportosítás előtt**.
- **HAVING:** Szűrés **csoportosítás után** (pl. aggregált értékekre).



# TSQL JOIN TYPES

Created by Steve Stedman

 <p>SELECT * FROM Table_1; SELECT * FROM Table_2;</p> <p>SELECT from two tables</p>	 <p>SELECT * FROM Table_1 t1 INNER JOIN Table_2 t2 ON t1.id = t2.fk;</p> <p>INNER JOIN</p>
 <p>SELECT * FROM Table_1 t1 LEFT JOIN Table_2 t2 ON t1.id = t2.fk;</p> <p>LEFT OUTER JOIN</p>	 <p>SELECT * FROM Table_1 t1 RIGHT JOIN Table_2 t2 ON t1.id = t2.fk;</p> <p>RIGHT OUTER JOIN</p>
 <p>SELECT * FROM Table_1 t1 WHERE EXISTS (SELECT 1 FROM Table_2 t2 WHERE t2.id = t1.fk );</p> <p>SEMI JOIN – Similar to INNER JOIN, with less duplication from Table 2.</p>	 <p>SELECT * FROM Table_1 t1 WHERE NOT EXISTS (SELECT 1 FROM Table_2 t2 WHERE t2.id = t1.fk );</p> <p>ANTI SEMI JOIN</p>
 <p>SELECT * FROM Table_1 t1 LEFT JOIN Table_2 t2 ON t1.id = t2.fk WHERE t2.fk is null;</p> <p>LEFT OUTER JOIN with exclusion – replacement for a NOT IN</p>	 <p>SELECT * FROM Table_1 t1 RIGHT JOIN Table_2 t2 ON t1.id = t2.fk WHERE t1.id is null;</p> <p>RIGHT OUTER JOIN with exclusion – replacement for a NOT IN</p>
 <p>SELECT * FROM Table_1 t1 FULL OUTER JOIN Table_2 t2 ON t1.id = t2.fk;</p> <p>FULL OUTER JOIN</p>	 <p>SELECT * FROM Table_1 t1 CROSS JOIN Table_2 t2;</p> <p>CROSS JOIN, like a FULL OUTER JOIN with out specifying JOIN condition.</p>
 <p>SELECT * FROM Table_1 t1 FULL OUTER JOIN Table_2 t2 ON t1.id = t2.fk WHERE t1.id is null OR t2.fk is null;</p> <p>FULL OUTER JOIN with exclusion – replacement for a double NOT IN</p>	 <p>SELECT * FROM Table_1 t1 INNER JOIN Table_2 t2 ON t1.id = t2.fk INNER JOIN Table_3 t3 ON t1.id = t3.fk;</p> <p>Two INNER JOINS</p>
 <p>SELECT * FROM Table_1 t1 FULL OUTER JOIN Table_2 t2 ON t1.id = t2.fk FULL OUTER JOIN Table_3 t3 ON t1.id = t3.fk;</p> <p>Two FULL OUTER JOINS</p>	 <p>SELECT * FROM Table_1 t1 INNER JOIN Table_2 t2 ON t1.id = t2.fk LEFT OUTER JOIN Table_3 t3 ON t1.id = t3.fk;</p> <p>INNER JOIN and a LEFT OUTER JOIN</p>
 <p>SELECT * FROM Table_1 t1 LEFT JOIN Table_2 t2 ON t1.id = t2.fk LEFT JOIN Table_3 t3 ON t1.id = t3.fk;</p> <p>TWO LEFT OUTER JOINS</p>	