



# Lesson 28

24.10.2024

```
public class Test1 {  
    public static void main(String[] args) {  
        int sum = 0;  
        for (int i = 0, j = 0; i < 6 & j < 5; ++i, j = i + 1) {  
            sum = +i;  
            System.out.println(sum);  
        }  
    }  
}
```

```
public class Test2 {  
    public static void main(String[] args) {  
        int arr[] = {11, 22, 33};  
        for (int i = 0; i < arr.length; i++)  
            System.out.println(arr[i] + " ");  
  
        int arrr[] = new int[3];  
        arrr[] = {11, 22, 33};  
        for (int i = 0; i < arrr.length; i++)  
            System.out.println(arrr[i] + " ");  
    }  
}
```



```
public class Test3 {  
    public static void main(String[] args) {  
        int[][] arr1 = new int[2][3];  
        int[][] arr2 = new int[2][];  
        int[][] arr3 = new int[][];  
        int[][] arr4 = new int[][3];  
    }  
}
```


```
public class Test4 {  
    public static void main(String[] args) {  
        for (int i = 0; i < 1; System.out.println("Java")) {  
            System.out.println("Scala");  
        }  
    }  
}
```



5 java questions



1. Відмінності String/StringBuilder/StringBuffer
2. Interface vs Abstract Class
3. override vs overload
4. Регіони пам'яті в JVM
5. java.util.collection.



```
ALTER TABLE table_name  
ADD column_name datatype;
```

```
INSERT INTO table_name (column1, column2, column3, ...)  
VALUES (value1, value2, value3, ...);
```

```
UPDATE table_name  
SET column1 = value1, column2 = value2, ...  
WHERE condition;
```

```
SELECT column1, column2, ...  
FROM table_name;
```

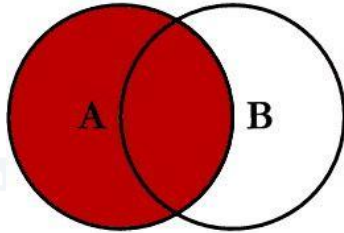
```
SELECT COUNT(column_name)  
FROM table_name  
WHERE condition;
```

```
SELECT AVG(column_name)  
FROM table_name  
WHERE condition;
```

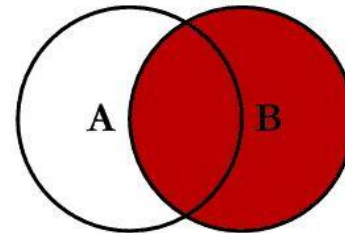
```
SELECT SUM(column_name)  
FROM table_name  
WHERE condition;
```



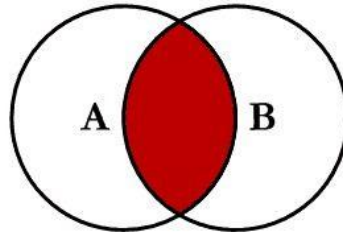
# SQL JOINS



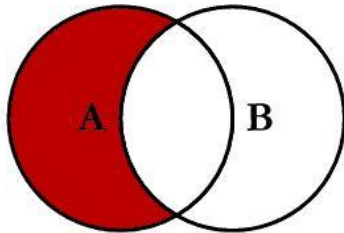
```
SELECT <select_list>  
FROM TableA A  
LEFT JOIN TableB B  
ON A.Key = B.Key
```



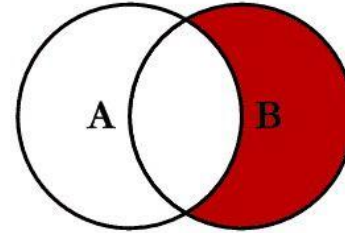
```
SELECT <select_list>  
FROM TableA A  
RIGHT JOIN TableB B  
ON A.Key = B.Key
```



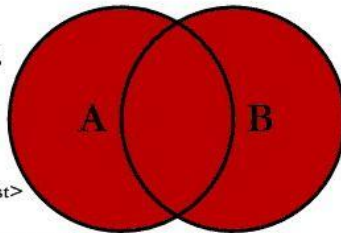
```
SELECT <select_list>  
FROM TableA A  
INNER JOIN TableB B  
ON A.Key = B.Key
```



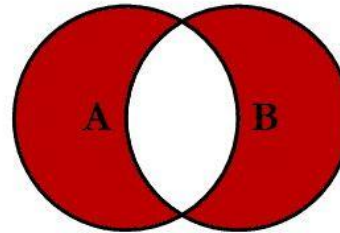
```
SELECT <select_list>  
FROM TableA A  
LEFT JOIN TableB B  
ON A.Key = B.Key  
WHERE B.Key IS NULL
```



```
SELECT <select_list>  
FROM TableA A  
RIGHT JOIN TableB B  
ON A.Key = B.Key  
WHERE A.Key IS NULL
```



```
SELECT <select_list>  
FROM TableA A  
FULL OUTER JOIN TableB B  
ON A.Key = B.Key
```



```
SELECT <select_list>  
FROM TableA A  
FULL OUTER JOIN TableB B  
ON A.Key = B.Key  
WHERE A.Key IS NULL  
OR B.Key IS NULL
```

# Employee

EmployeeID	Ename	DeptID	Salary
1001	John	2	4000
1002	Anna	1	3500
1003	James	1	2500
1004	David	2	5000
1005	Mark	2	3000
1006	Steve	3	4500
1007	Alice	3	3500

**SELECT** DeptID, AVG(Salary)  
**FROM** Employee  
**GROUP BY** DeptID;

**GROUP BY**  
Employee Table  
using DeptID

DeptID	AVG(Salary)
1	3000.00
2	4000.00
3	4250.00



## Aggregate Functions

**SUM()**: Returns the sum or total of each group.

**COUNT()**: Returns the number of rows of each group.

**AVG()**: Returns the average and mean of each group.

**MIN()**: Returns the minimum value of each group.

**MAX()**: Returns the maximum value of each group.

## Employee

EmployeeID	Ename	DeptID	Salary
1001	John	2	4000
1002	Anna	1	3500
1003	James	1	2500
1004	David	2	5000
1005	Mark	2	3000
1006	Steve	3	4500
1007	Alice	3	3500

```
SELECT DeptID, AVG(Salary)
FROM Employee
GROUP BY DeptID;
```

GROUP BY  
Employee Table  
using DeptID

DeptID	AVG(Salary)
1	3000.00
2	4000.00
3	4250.00

```
SELECT DeptID, AVG(Salary)
FROM Employee
GROUP BY DeptID
HAVING AVG(Salary) > 3000;
```

HAVING

DeptID	AVG(Salary)
2	4000.00
3	4250.00

## Primary key (PK)

У кожній таблиці БД може бути первинний ключ. Під первинним ключем розуміють поле або набір полів, що однозначно (унікально) ідентифікують запис. Первинний ключ має бути мінімально достатнім: він має складатися з полів, видалення яких із первинного ключа не позначиться з його унікальності.

## Foreign key(FK)

Забезпечує однозначний логічний зв'язок між таблицями однієї БД.

