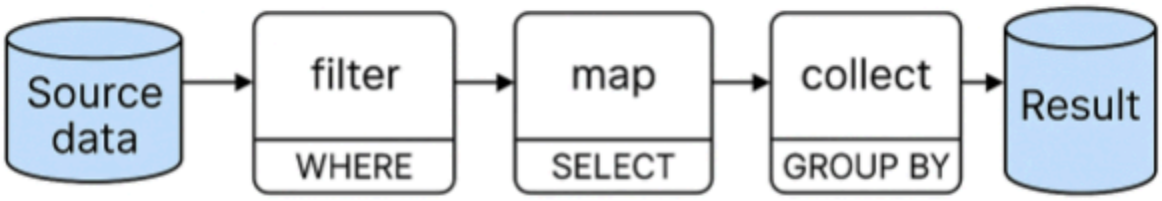
Stream API



🧠 **Understanding Stream API in Terms of SQL**
Think of the **Stream API** as a way to **query data** from Java collections just like how **SQL queries** data from databases.

🔹 1. **Filtering Data**[1]

SQL Query:

```
SELECT * FROM Employee WHERE salary > 50000;
```

Stream API:

```
employees.stream()
    .filter(e -> e.getSalary() > 50000)
    .collect(Collectors.toList());
```

🔹 **2. Selecting Specific Fields (Mapping)**

SQL Query:

```
SELECT name FROM Employee;
```

Stream API:+

```
employees.stream()
    .map(Employee::getName)
    .collect(Collectors.toList());
```

[1]Created By Sandip Varagle

🔹 3. Aggregating Data (like COUNT, AVG, etc.)

SQL Query:

```
SELECT COUNT(*) FROM Employee;
```

[2]

Stream API:

```
long count = employees.stream().count();
```

🔹 4. Grouping Data

SQL Query:

```
SELECT department, COUNT(*) FROM Employee GROUP BY
department;
```

Stream API:

```
Map<String, Long> groupByDept =
 employees.stream()
     .collect(Collectors.groupingBy(Employee::getDepartment,
Collectors.counting()));
```

🔹 5. Sorting Data

SQL Query:

```
SELECT * FROM Employee ORDER BY salary DESC;
```

Stream API:

```
employees.stream()
.sorted(Comparator.comparing(Employee::getSalary).reversed())
     .collect(Collectors.toList());
```

🔹 6.Combination of filter and Sort

SQL Query:

```
SELECT name, age FROM users WHERE age > 25 ORDER BY age;
```

Stream API:

```
List<User> filteredUsers = users.stream()
                        .filter(user -> user.getAge() > 25)
```

[2] Created By Sandip Vargale

[3]

```
        .sorted(Comparator.comparingInt(User::getAge))
        .collect(Collectors.toList());
```

🔹 SQL vs. Stream API - Comparison Table 🔹

Operation	SQL Equivalent	Java Stream API Example
Filtering	WHERE condition	list.stream().filter(x -> x > 10)
Mapping	SELECT column_name	list.stream().map(x -> x * 2)
Sorting	ORDER BY column_name	list.stream().sorted()
Aggregation	SUM(), COUNT(), AVG()	list.stream().reduce()
Grouping	GROUP BY column_name	list.stream().collect(Collectors.groupingBy())
Limit Records	LIMIT N	list.stream().limit(N)
Joining Data	INNER JOIN / LEFT JOIN	Stream.concat(list1.stream(), list2.stream())

[3] Created By Sandip Vargale