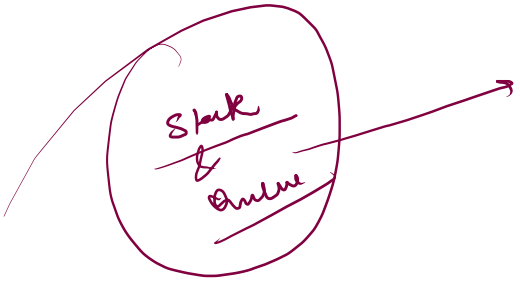
2-Most asked Interview Question

4 → SQL → Hackerrank

Implementation of a Text-Editor



↳ Write op^n

↳ Read op^n

↳ Undo op^n

↳ Redo op^n

⇒ 10-15 minutes

Write

Redo

Stack → 2 → ① Undo

↳ ② Redo

① Write → 'a' undo.push('a');

```java
public class Solution {
    static Stack<Character> undo = new Stack<>();
    static Stack<Character> redo = new Stack<>();

    public static void writeOperation(char x){
        undo.push(x);
    }

    public static void undoOperation(){
        if(undo.size() == 0){
            System.out.println("You can't perform undo operation");
            return;
        }
        redo.push(undo.pop());
    }

    public static void redoOperation(){
        if(redo.size() == 0){
            System.out.println("You can't perform redo operation");
            return;
        }
        undo.push(redo.pop());
    }

    public static void readOperation(){
        Stack<Character> read = new Stack<>();
        while(undo.size()>0){
            System.out.println(undo.peek());
            read.push(undo.pop());
        }
        System.out.print("Read Operation starts: ");

        while(read.size()>0){
            System.out.print(read.peek()+" ");
            undo.push(read.pop());

        }

        System.out.println("Read operation completed");

    }

    public static void callFunction(String data[]){
       for(int i=0;i<data.length;i++){
            if (data[i].equals("READ")){
                readOperation();

            }
            else if(data[i].equals("UNDO")){
                undoOperation();

            }else if(data[i].equals("REDO")){
                redoOperation();

            }else{
                writeOperation(data[i].charAt(6));

    public static void main(String [] args){
            String data[] = {"WRITE A","WRTIE B","READ","UNDO","WRITE C","REDO","READ"};
            callFunction(data);
    }
}
```

```java
import java.io.*;
import java.util.*;

public class Main {
    static Stack<Character> undo = new Stack<>();
    static Stack<Character> redo = new Stack<>();

    public static void writeOperation(char x){
        undo.push(x);
    }

    public static void undoOperation(){
        if(undo.size() == 0){
            System.out.println("You can't perform undo operation");
            return;
        }
        redo.push(undo.pop());
    }

public static void redoOperation(){
    if(redo.size() == 0){
        System.out.println("You can't perform redo operation");
        return;
    }
    undo.push(redo.pop());
}

    public static void main(String [] args){
        Scanner scn = new Scanner(System.in);

        while(true){
            String operation = scn.next();
            if( operation.equals("false")){
                break;
            }else{
                callFunction(operation);
            }
        }

}

public static void readOperation(){
    Stack<Character> read = new Stack<>();
    while(undo.size()>0){
        read.push(undo.pop());
    }
    System.out.println("Read Operation starts: ");

    while(read.size()>0){
        System.out.print(read.peek()+" ");
        undo.push(read.pop());

    }
    System.out.println();

    System.out.println("Read operation completed");

}

public static void callFunction(String data){
   if (data.equals("READ")){
       readOperation();

   }
   else if(data.equals("UNDO")){
       undoOperation();

   }else if(data.equals("REDO")){
       redoOperation();

   }else if (data.length() > 5 && data.substring(0,5).equals("WRITE")){

       writeOperation(data.charAt(5));

   }else {
       System.out.println("Invalid Operation");
   }
}
```

5-10 minutes

Random → class

Write a program generate random Password

D.S.A

- ↳ 1 Capital letter
- ↳ 1 small letter
- ↳ 1 digit
- ↳ 1 special characters → !, @ # $ % ^ & *
- ↳ length >= 8

Random

1 CL, 1 SL, 1 digit, 1 special.

1 capital ✓
1 small ✓
1 digit ✓
1 special ✓

next Int

↳ String capitalLetters = "ABCDEFGHIJKLMNOPQRSTUVWXYZ" (0, 26)

↳ String smallLetters = "abc ... "

↳ String digits = "123456789"

↳ String specialCharacters = "!@#$%^&*"

↳ String combinedCharacters = capitalLetters + smallLetters + digits + specialCharacters;

Random random = new Random();

↳ String Password = "";

Password += capitalLetters.charAt(random.nextInt(capitalLetters.length())); (5)

capitalLetters.charAt(5); → F

```java
public class Solution {
   public static String generateRandomPassword(int length){

       if(length<8){
           return "Password length cannot be less than 8 characters";
       }

       String specialCharacters = "!@#$%^&*";
       String combinedLetters ="ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890abcdefghijklmnopqrstuvwxyz!@#$%^&*";

       String password = "";
       Random random = new Random();

       password+=(char)(random.nextInt(26)+'A');
       password+=(char)(random.nextInt(26)+'a');
       password+=random.nextInt(10);
       password+=specialCharacters.charAt(random.nextInt(specialCharacters.length()));

       for(int i=4;i<=length;i++){
           password+=combinedLetters.charAt(random.nextInt(combinedLetters.length()));
       }


       return password;
   }


    public static void main(String [] args){

        System.out.println(generateRandomPassword(16));


    }

}
```

System → codingshuff →

git repository → git-demo → clone

clone

1

git clone + url

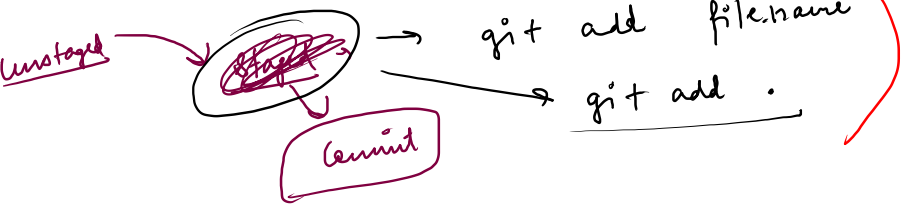
2 minute

git-demo

↳ demo.java

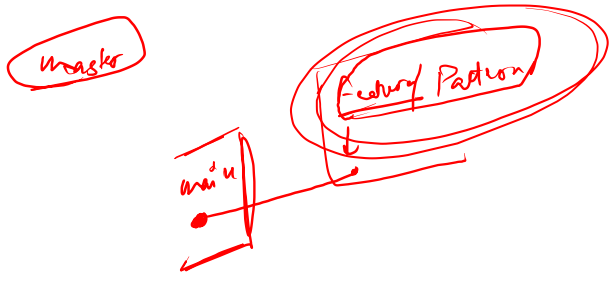
↳ printing.java

→ git status



git restore --staged filename

10-15



Commit

↳ git commit -m "message"

git checkout -b "branch-name"

↳ Create Branch ↳ Program ↳ commit & push.

↳ main PR creat ↳ approve

```
karti@NU14A1 MINGW64 ~/OneDrive/Desktop/git-demo (main)
$ git checkout -b feature/Pattern
Switched to a new branch 'feature/Pattern'
```

```
karti@NU14A1 MINGW64 ~/OneDrive/Desktop/git-demo (feature/Pattern)
$ git add .

karti@NU14A1 MINGW64 ~/OneDrive/Desktop/git-demo (feature/Pattern)
$ git status
On branch feature/Pattern
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   Pattern1.java


karti@NU14A1 MINGW64 ~/OneDrive/Desktop/git-demo (feature/Pattern)
$ git commit -m "Pattern 1 code"
[feature/Pattern 47236bd] Pattern 1 code
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 Pattern1.java

karti@NU14A1 MINGW64 ~/OneDrive/Desktop/git-demo (feature/Pattern)
$ git push -u origin feature/Pattern
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
```

→ Create branch from main.

→ Program.

→ add, Commit, Push.

→ PR for main branch

↳ Create Branch ↳ Program ↳ commit & push.

↳ main PR creat ↳ approve

Condition (1<9a8a2 ?)

i>=1 && i<=9

1-9

0-9

A2

az

- → git clone url / git init
- → git status
- → git add filename / git add . (for multiple files)
- → git restore --staged file name
- → git commit -m " message "
- → git push -u origin branch-name
- → git checkout -b branch-name
- → git checkout branch-name
- → PR (pull request)

String → Group of characters

str = 0 1 2 3 4 5 6
Geekster

→ " "

→ string-name.length() ✓

→ String-name[index];

→ for (int i = 0; i < string-name; i++)

{ sysos (string-name.charAt(i));

}

→ 0-based indexing

→ array length

4+1

→ str.substring(2); → ekster

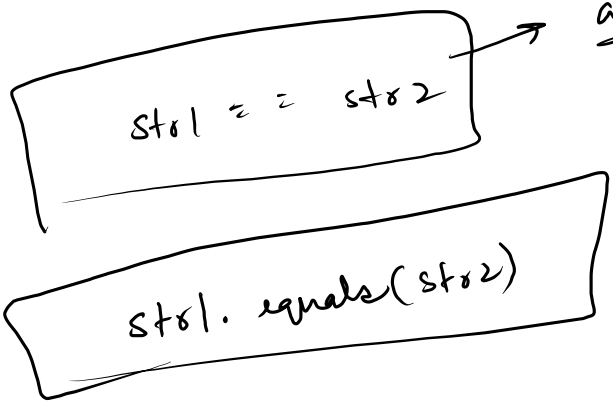
→ str.substring(2, 5) → eks
(ending index+1)

→ sc.next() → Geekster coding → o/p → Geekster

→ sc.nextline() → o/p → Geekster coding

str1 = Geekster

str2 = Geekster




```
if (str.length() == 0)
    return 0;
for (int i = j; i < k; i++) {
    if (str.charAt(i) == ' ')
        count++;
}
return count;
```

ab

db

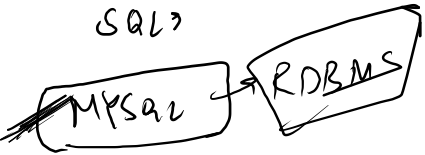
abc — — bcd

— — — — —

— remove starting space.

→ remove ending space.

SQL?



Programming Language → C++, Java, Python

orderBy DOJ

order by DOJ desc

Select * from Employee data ;

Select * from Employee data where City = 'Aligarh'

Delete

Employee data

EmpID	EmpName	Designation	DOJ	City
121	Ravi	SE	April 2021	Aligarh
141	Akash	Manager	Jan 2022	Bangalore
151	Abhinav	SE	Jan 2022	Jaipur

with Manager Designation

DOJ desc

141 Akash Manager Jan 2022 Bangalore
151 Abhinav SE Jan 2021 Jaipur
121 Ravi SE April 2021 Aligarh
131 Ram SSE Oct 2020 Aligarh

select city from Employee data

Aligarh
Aligarh
Bangalore

select distinct city from Employee data

Aligarh
Bangalore

EmpID → select EmpID from Employee data

121
131
141

select * from Employee Data where City = 'Aligarh';

121	Ravi	SE	April 2021	Aligarh
131	Ram	SSE	Oct 2020	Aligarh

Insert into table_name (EmpID, EmpName)

<>

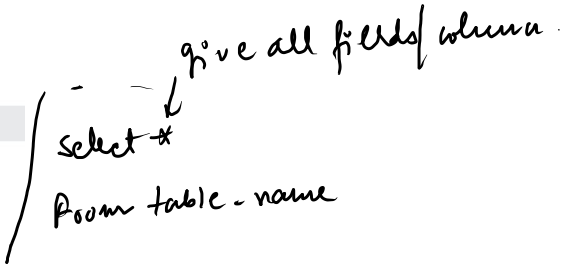
select * from Employee Data where EmpID = 121

The SQL SELECT Statement

The `SELECT` statement is used to select data from a database.

SELECT Syntax

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



Select

The SQL SELECT DISTINCT Statement

The `SELECT DISTINCT` statement is used to return only distinct (different) values.

Inside a table, a column often contains many duplicate values; and sometimes you only want to list the different (distinct) values.

SELECT DISTINCT Syntax

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

The SQL WHERE Clause

The `WHERE` clause is used to filter records.

It is used to extract only those records that fulfill a specified condition.

WHERE Syntax

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

Note: The `WHERE` clause is not only used in `SELECT` statements, it is also used in `UPDATE`, `DELETE`, etc.!

The SQL AND, OR and NOT Operators

The `WHERE` clause can be combined with `AND`, `OR`, and `NOT` operators.

The `AND` and `OR` operators are used to filter records based on more than one condition:

- The `AND` operator displays a record if all the conditions separated by `AND` are TRUE.
- The `OR` operator displays a record if any of the conditions separated by `OR` is TRUE.

The `NOT` operator displays a record if the condition(s) is NOT TRUE.

AND Syntax

```
SELECT column1, column2, ...
FROM table_name
WHERE condition1 AND condition2 AND condition3 ...;
```

OR Syntax

```
SELECT column1, column2, ...
FROM table_name
WHERE condition1 OR condition2 OR condition3 ...;
```

NOT Syntax

```
SELECT column1, column2, ...
FROM table_name
WHERE NOT condition;
```

The SQL ORDER BY Keyword

The `ORDER BY` keyword is used to sort the result-set in ascending or descending order.

The `ORDER BY` keyword sorts the records in ascending order by default. To sort the records in descending order, use the `DESC` keyword.

ORDER BY Syntax

```
SELECT column1, column2, ...
FROM table_name
ORDER BY column1, column2, ... ASC|DESC;
```

INSERT INTO Syntax

It is possible to write the `INSERT INTO` statement in two ways:

1. Specify both the column names and the values to be inserted:

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

2. If you are adding values for all the columns of the table, you do not need to specify the column names in the SQL query. However, make sure the order of the values is in the same order as the columns in the table. Here, the `INSERT INTO` syntax would be as follows:

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

SQL NULL Values

❮ Previous

Next ❯

What is a NULL Value?

A field with a NULL value is a field with no value.

If a field in a table is optional, it is possible to insert a new record or update a record without adding a value to this field. Then, the field will be saved with a NULL value.

How to Test for NULL Values?

It is not possible to test for NULL values with comparison operators, such as =, <, or <>.

We will have to use the `IS NULL` and `IS NOT NULL` operators instead.

IS NULL Syntax

```
SELECT column_names
FROM table_name
WHERE column_name IS NULL;
```

IS NOT NULL Syntax

```
SELECT column_names
FROM table_name
WHERE column_name IS NOT NULL;
```

The SQL UPDATE Statement

The `UPDATE` statement is used to modify the existing records in a table.

UPDATE Syntax

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

The SQL DELETE Statement

The `DELETE` statement is used to delete existing records in a table.

DELETE Syntax

```
DELETE FROM table_name WHERE condition;
```

Note: Be careful when deleting records in a table! Notice the `WHERE` clause in the `DELETE` statement. The `WHERE` clause specifies which record(s) should be deleted. If you omit the `WHERE` clause, all records in the table will be deleted!

The SQL SELECT TOP Clause

The `SELECT TOP` clause is used to specify the number of records to return.

The `SELECT TOP` clause is useful on large tables with thousands of records. Returning a large number of records can impact performance.

MySQL Syntax:

```
SELECT column_name(s)
FROM table_name
WHERE condition
LIMIT number;
```

The SQL MIN() and MAX() Functions

The `MIN()` function returns the smallest value of the selected column.

The `MAX()` function returns the largest value of the selected column.

MIN() Syntax

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

MAX() Syntax

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

COUNT() Syntax

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

SUM() Syntax

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

AVG() Syntax

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

The SQL LIKE Operator

The `LIKE` operator is used in a `WHERE` clause to search for a specified pattern in a column.

There are two wildcards often used in conjunction with the `LIKE` operator:

- The percent sign (%) represents zero, one, or multiple characters
- The underscore sign (_) represents one, single character

LIKE Syntax

```
SELECT column1, column2, ...
FROM table_name
WHERE columnN LIKE pattern;
```

where country = 'German' 'Australia'

where country in ('German' , 'Aust

LIKE Operator	Description
WHERE CustomerName LIKE 'a%'	Finds any values that start with "a"
WHERE CustomerName LIKE '%a'	Finds any values that end with "a"
WHERE CustomerName LIKE '%or%'	Finds any values that have "or" in any position
WHERE CustomerName LIKE '_r%'	Finds any values that have "r" in the second position
WHERE CustomerName LIKE 'a_%'	Finds any values that start with "a" and are at least 2 characters in length
WHERE CustomerName LIKE 'a__%'	Finds any values that start with "a" and are at least 3 characters in length
WHERE ContactName LIKE 'a%o'	Finds any values that start with "a" and ends with "o"

The SQL IN Operator

The `IN` operator allows you to specify multiple values in a `WHERE` clause.

The `IN` operator is a shorthand for multiple `OR` conditions.

IN Syntax

```
SELECT column_name(s)
FROM table_name
WHERE column_name IN (value1, value2, ...);
```

The SQL BETWEEN Operator

The `BETWEEN` operator selects values within a given range. The values can be numbers, text, or dates.

The `BETWEEN` operator is inclusive: begin and end values are included.

BETWEEN Syntax

```
SELECT column_name(s)
FROM table_name
WHERE column_name BETWEEN value1 AND value2;
```

SQL Aliases

SQL aliases are used to give a table, or a column in a table, a temporary name.

Aliases are often used to make column names more readable.

An alias only exists for the duration of that query.

An alias is created with the `AS` keyword.

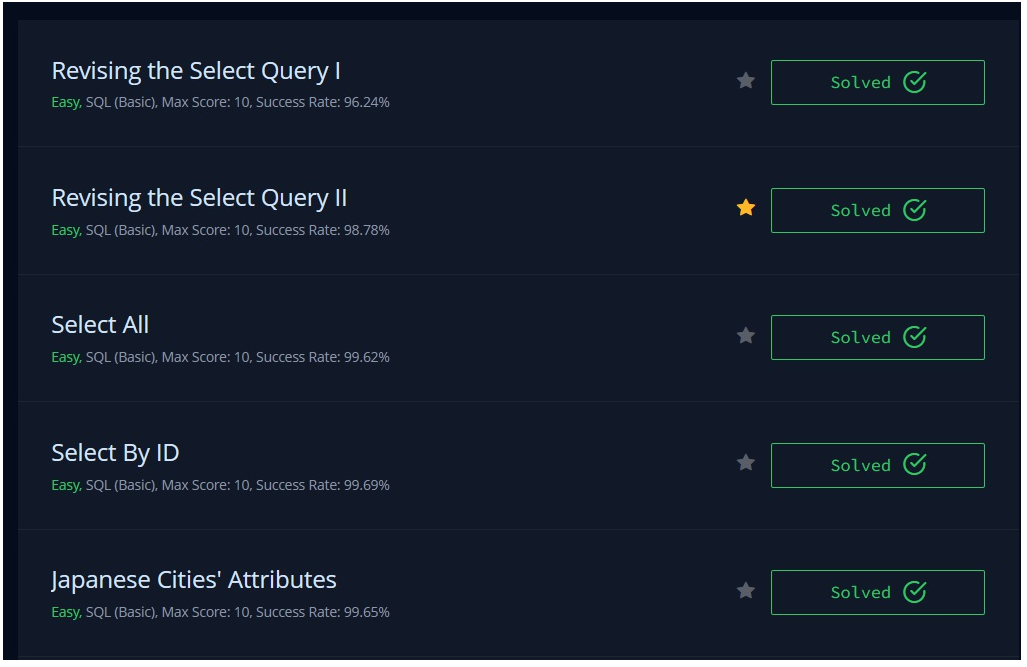
Alias Column Syntax

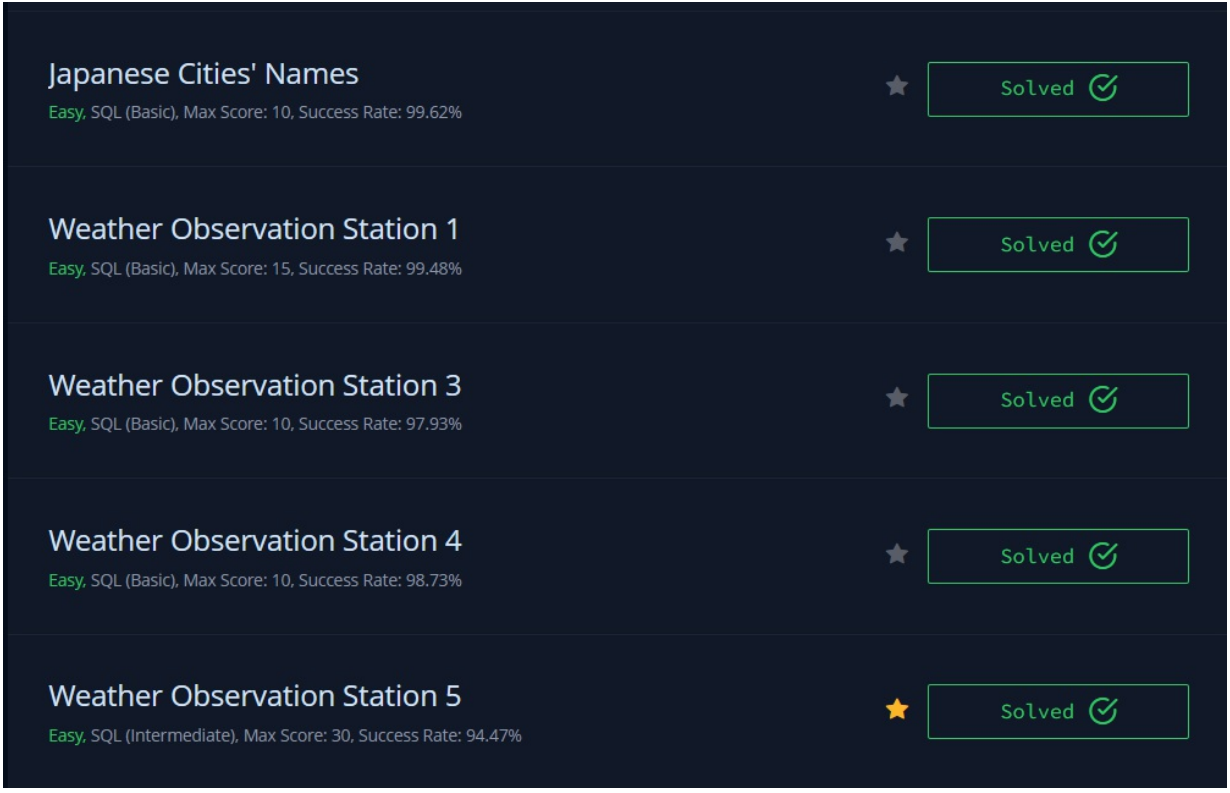
```
SELECT column_name AS alias_name
FROM table_name;
```

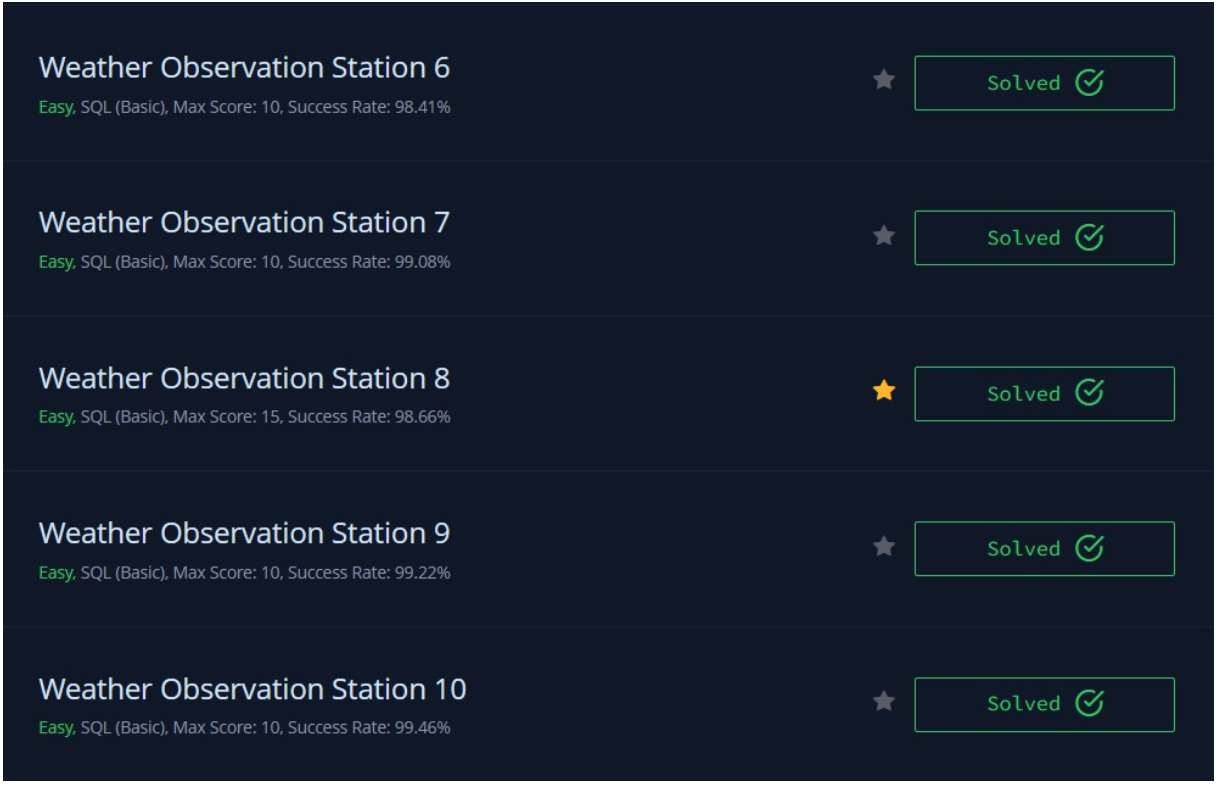
Alias Table Syntax

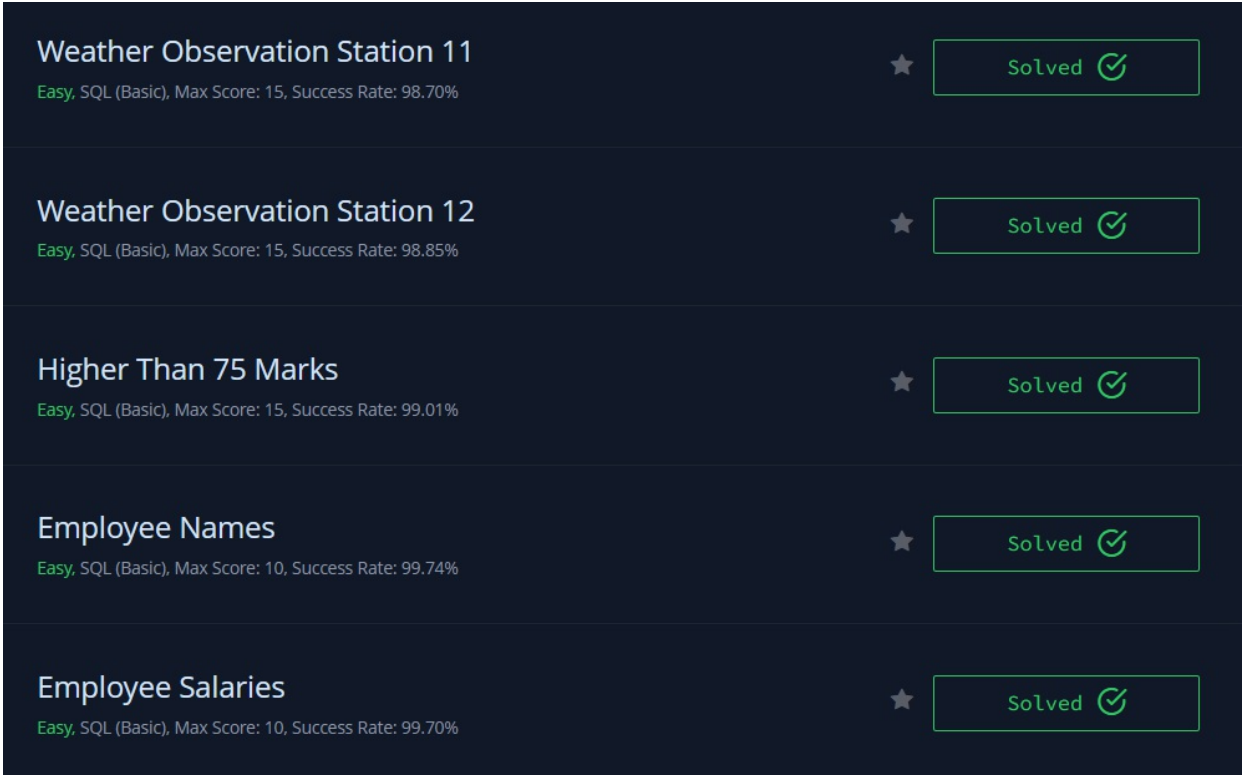
```
SELECT column_name(s)
FROM table_name AS alias_name;
```

15 minute


Revising the Select Query I
Easy, SQL (Basic), Max Score: 10, Success Rate: 96.24%
Solved
Revising the Select Query II
Easy, SQL (Basic), Max Score: 10, Success Rate: 98.78%
Solved
Select All
Easy, SQL (Basic), Max Score: 10, Success Rate: 99.62%
Solved
Select By ID
Easy, SQL (Basic), Max Score: 10, Success Rate: 99.69%
Solved
Japanese Cities' Attributes
Easy, SQL (Basic), Max Score: 10, Success Rate: 99.65%
Solved


Japanese Cities' Names
Easy, SQL (Basic), Max Score: 10, Success Rate: 99.62%
Solved
Weather Observation Station 1
Easy, SQL (Basic), Max Score: 15, Success Rate: 99.48%
Solved
Weather Observation Station 3
Easy, SQL (Basic), Max Score: 10, Success Rate: 97.93%
Solved
Weather Observation Station 4
Easy, SQL (Basic), Max Score: 10, Success Rate: 98.73%
Solved
Weather Observation Station 5
Easy, SQL (Intermediate), Max Score: 30, Success Rate: 94.47%
Solved


Weather Observation Station 6
Easy, SQL (Basic), Max Score: 10, Success Rate: 98.41%
Solved
Weather Observation Station 7
Easy, SQL (Basic), Max Score: 10, Success Rate: 99.08%
Solved
Weather Observation Station 8
Easy, SQL (Basic), Max Score: 15, Success Rate: 98.66%
Solved
Weather Observation Station 9
Easy, SQL (Basic), Max Score: 10, Success Rate: 99.22%
Solved
Weather Observation Station 10
Easy, SQL (Basic), Max Score: 10, Success Rate: 99.46%
Solved


Weather Observation Station 11
Easy, SQL (Basic), Max Score: 15, Success Rate: 98.70%
Solved
Weather Observation Station 12
Easy, SQL (Basic), Max Score: 15, Success Rate: 98.85%
Solved
Higher Than 75 Marks
Easy, SQL (Basic), Max Score: 15, Success Rate: 99.01%
Solved
Employee Names
Easy, SQL (Basic), Max Score: 10, Success Rate: 99.74%
Solved
Employee Salaries
Easy, SQL (Basic), Max Score: 10, Success Rate: 99.70%
Solved

Revising Aggregations - The Count Function

Easy, SQL (Basic), Max Score: 10, Success Rate: 99.56%

Solved

Revising Aggregations - The Sum Function

Easy, SQL (Basic), Max Score: 10, Success Rate: 99.64%

Solved

Revising Aggregations - Averages

Easy, SQL (Basic), Max Score: 10, Success Rate: 99.79%

Solved

Average Population

Easy, SQL (Basic), Max Score: 10, Success Rate: 99.60%

Solved

Japan Population

Easy, SQL (Basic), Max Score: 10, Success Rate: 99.85%

Solved

Population Density Difference

Easy, SQL (Basic), Max Score: 10, Success Rate: 99.85%

Solved

The Blunder

Easy, SQL (Basic), Max Score: 15, Success Rate: 97.94%

Solved

Top Earners

Easy, SQL (Basic), Max Score: 20, Success Rate: 98.27%

Solved

Weather Observation Station 2

Easy, SQL (Basic), Max Score: 15, Success Rate: 99.20%

Solved

Weather Observation Station 13

Easy, SQL (Basic), Max Score: 10, Success Rate: 99.42%

Solved

Weather Observation Station 14

Easy, SQL (Basic), Max Score: 10, Success Rate: 99.63%

Solved

Weather Observation Station 15

Easy, SQL (Basic), Max Score: 15, Success Rate: 98.98%

Solved

Weather Observation Station 16

Easy, SQL (Basic), Max Score: 10, Success Rate: 99.67%

Solved

Weather Observation Station 17

Easy, SQL (Basic), Max Score: 15, Success Rate: 99.30%

Solved

Weather Observation Station 18

Medium, SQL (Basic), Max Score: 25, Success Rate: 98.75%

Solved

Joins :-

Employee

Emp-Id	Name	DOJ
1	Ram	22nd April 2021
2	Ravi	24th March 2022
3	Shyam	19th Sep, 2020
4	Akash	14th Nov, 2021

Department

id	Dept.	Salary
1	HR	25000
2	IT Support	35000
4	EPA	50000

Select salary from Department where Dept = 'HR'

Employee → data 24th Nov, 2021

Select * from Employee where DOJ > 19 Nov 2021

① Cross Product

② Condition

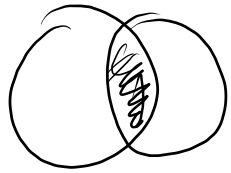
At least one column data same

m , n

→ m * n

Joins

Cross Product



Joins

Employee Name → Dept → 'HR' → Select Name from Employee , Department where Employee.Emp-id = Department.id

Emp-id	Name	DOJ	id	Dept.	Salary
1	Ram	22nd April 2021	1	HR	25000
1	Ram	22nd April 2021	2	IT Support	35000
1	Ram	22nd April 2021	4	EPA	50000
2	Ravi	24th March 2022	1	HR	25000
2	Ravi	24th March 2022	2	IT Support	35000
2	Ravi	24th March 2022	4	EPA	50000

3	Shyam	19th Sep 2020	1	HR	25000
3	Shyam	19th Sep 2020	2	IT Support	35000
3	Shyam	19th Sep 2020	4	EPA	50000
4	Akash	14th Nov 2021	1	HR	25000
4	Akash	14th Nov 2021	2	IT Support	35000
4	Akash	14th Nov 2021	4	EPA	50000

Below is a selection from the "Orders" table:

OrderID	CustomerID	EmployeeID	OrderDate	ShipperID
10308	2	7	1996-09-18	3
10309	37	3	1996-09-19	1
10310	77	8	1996-09-20	2

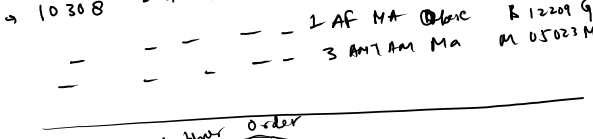
And a selection from the "Customers" table:

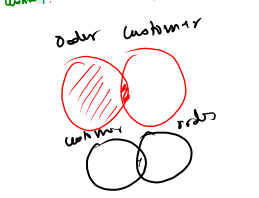
CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico

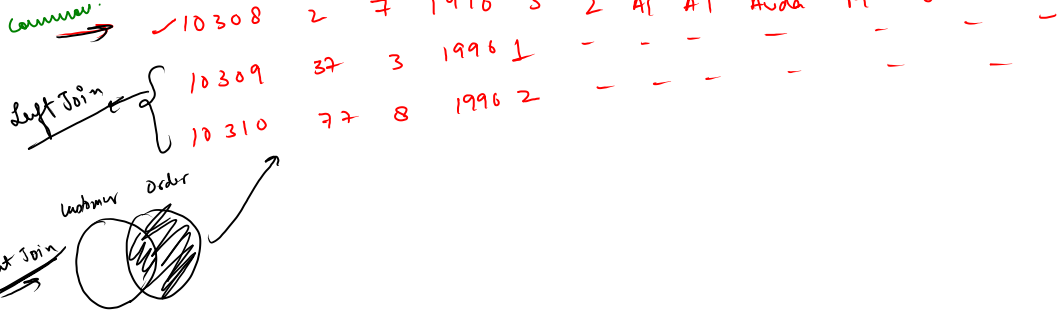
Example

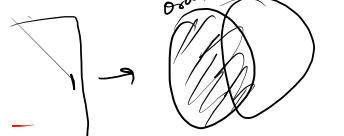
```
SELECT Customers.CustomerName, Orders.OrderID
FROM Customers
LEFT JOIN Orders ON Customers.CustomerID = Orders.CustomerID
ORDER BY Customers.CustomerName;
```

OI	CID	EID	OD	S ID	CID	CN	CON	Add	City	PC	Country
10308	2	7	1996	3	1	Af	MA	Obere	B	12209	G
〃	〃	〃	〃	〃	2	AT	AT	Avda	M	05021	M
〃	〃	〃	〃	〃	3	AMT	AM	Ma	M	05023	M
10309	37	3	1998	1	1	Af	MA	Obre	B	12209	G
〃	〃	〃	〃	〃	2	AT	AT	Avda	M	05021	M
〃	〃	〃	〃	〃	3	AMT	AM	Ma	M	05023	M
10310	77	8	1996	2	1	Af	MA	Obre	B	12209	G
〃	〃	〃	〃	〃	2	AT	AT	Avda	M	05021	M
〃	〃	〃	〃	〃	3	AMT	AM	Ma	M	05023	M









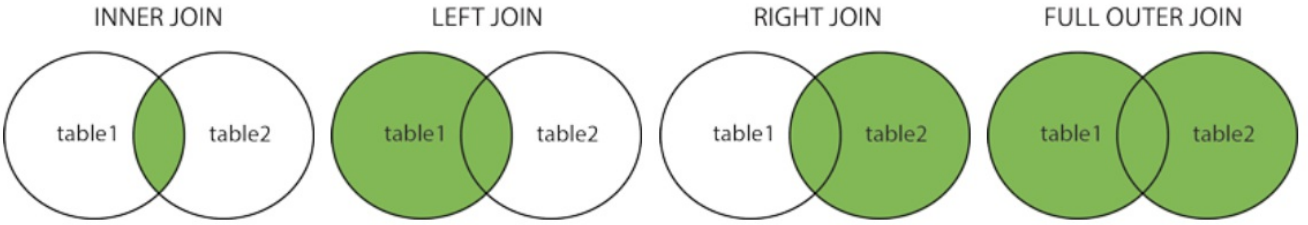
SQL JOIN

A `JOIN` clause is used to combine rows from two or more tables, based on a related column between them.

Different Types of SQL JOINs

Here are the different types of the JOINs in SQL:

- `(INNER) JOIN`: Returns records that have matching values in both tables
- `LEFT (OUTER) JOIN`: Returns all records from the left table, and the matched records from the right table
- `RIGHT (OUTER) JOIN`: Returns all records from the right table, and the matched records from the left table
- `FULL (OUTER) JOIN`: Returns all records when there is a match in either left or right table



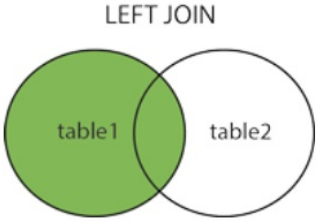
SQL LEFT JOIN Keyword

The `LEFT JOIN` keyword returns all records from the left table (table1), and the matching records from the right table (table2). The result is 0 records from the right side, if there is no match.

LEFT JOIN Syntax

```
SELECT column_name(s)
FROM table1
LEFT JOIN table2
ON table1.column_name = table2.column_name;
```

Note: In some databases LEFT JOIN is called LEFT OUTER JOIN.



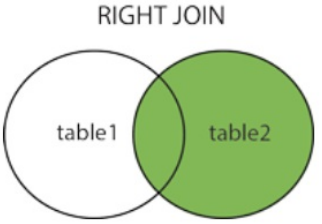
SQL RIGHT JOIN Keyword

The `RIGHT JOIN` keyword returns all records from the right table (table2), and the matching records from the left table (table1). The result is 0 records from the left side, if there is no match.

RIGHT JOIN Syntax

```
SELECT column_name(s)
FROM table1
RIGHT JOIN table2
ON table1.column_name = table2.column_name;
```

Note: In some databases `RIGHT JOIN` is called `RIGHT OUTER JOIN`.



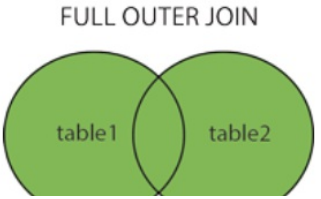
SQL FULL OUTER JOIN Keyword

The `FULL OUTER JOIN` keyword returns all records when there is a match in left (table1) or right (table2) table records.

Tip: `FULL OUTER JOIN` and `FULL JOIN` are the same.

FULL OUTER JOIN Syntax

```
SELECT column_name(s)
FROM table1
FULL OUTER JOIN table2
ON table1.column_name = table2.column_name
WHERE condition;
```



Full Join

NAME

ID	NAME
1	A
2	B

DEPT

DID	DEPT
1	HR
3	EPA

→

ID	NAME	DID	DEPT
1	A	1	HR.
2	B	—	—
—	—	3	EPA

SQL Self Join

‹ Previous

SQL Self Join

A self join is a regular join, but the table is joined with itself.

Self Join Syntax

```
SELECT column_name(s)
FROM table1 T1, table1 T2
WHERE condition;
```

T1 and *T2* are different table aliases for the same table.

Self Join

Course

SID	C-ID	Duration
S1	C1	1Y
S2	C2	2Y
S1	C2	1.5Y

Find student id who is enrolled at least 2 course

SID	CID	Duration	SID	CID	Duration
S1	C1	1Y	S1	C1	1Y
S1	C1	1Y	S2	C2	2Y
S1	C1	1Y	S1	C2	1.5Y
S2	C2	2Y	S1	C1	1Y
S2	C2	2Y	S2	C2	2Y
S2	C2	2Y	S1	C2	1.5Y
S1	C2	1.5Y	S1	C1	1Y
S1	C2	1.5Y	S2	C2	2Y
S1	C2	1.5Y	S1	C2	1.5Y

```
Select DISTINCT (CO1.SID)
from Course as CO1, Course as CO2
where CO1.SID = CO2.SID
and CO1.CID <> CO2.CID
```

⇒ S1

↳ No. of parameters

↳ Datatypes

↳ Same order occurrence

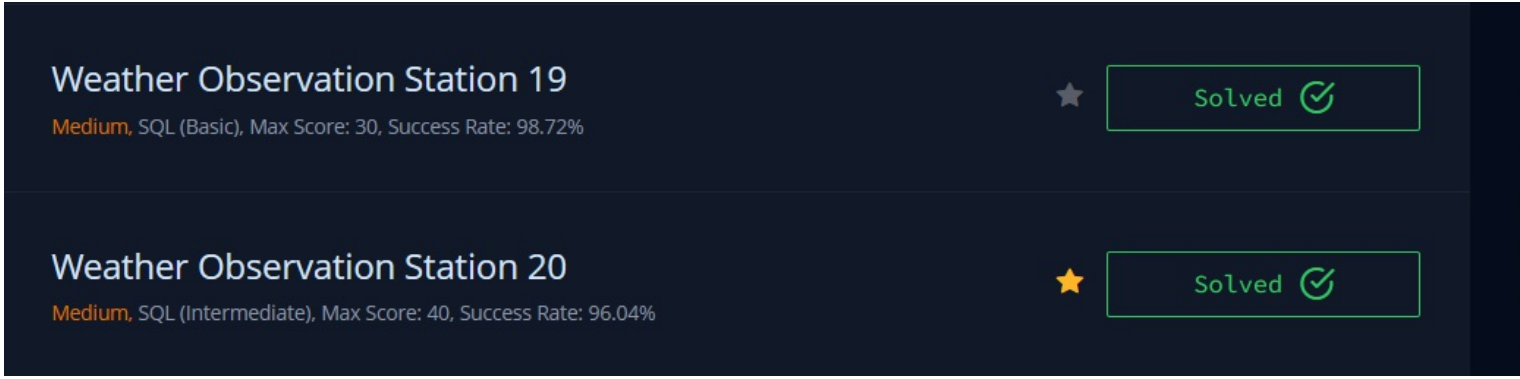
The SQL UNION Operator

The `UNION` operator is used to combine the result-set of two or more `SELECT` statements.

- Every `SELECT` statement within `UNION` must have the same number of columns
- The columns must also have similar data types
- The columns in every `SELECT` statement must also be in the same order

UNION Syntax

```
SELECT column_name(s) FROM table1
UNION
SELECT column_name(s) FROM table2;
```


Weather Observation Station 19
Medium, SQL (Basic), Max Score: 30, Success Rate: 98.72%
Solved
Weather Observation Station 20
Medium, SQL (Intermediate), Max Score: 40, Success Rate: 96.04%
Solved

GROUP BY →

Sum :→ Population

↳ Country Wise

Population	Country
1	JPN
2	India
3	CHINA
4	Australia
2	JPN
5	CHINA

→ DATA

Select sum (Population) from DATA GROUP BY country.

3

2

8

4

Last Week :-

- Git Interview Questions
- SQL Interview Questions
- String memory mapping, Hashmap implementation
- Practice Questions → Leetcode | GFG | ——
- OOP's | Trees