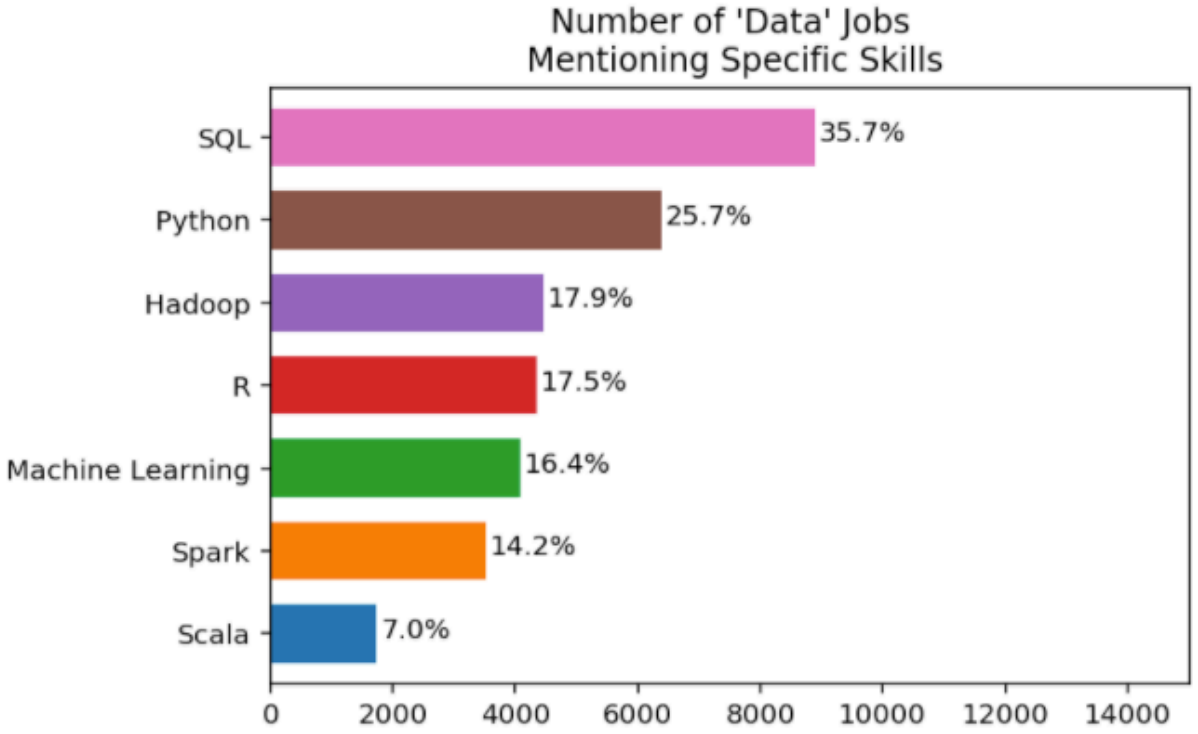
PART 5: Ace SQL Interviews



Source: Indeed

SQL is in demand.

PART 5: Ace SQL Interviews

Interview Q1:

What is DBMS and what are the different types of DBMS?

PART 5: Ace SQL Interviews

Interview Q1:

What is DBMS and what are the different types of DBMS?

A Database management system (DBMS) is a software application that interacts with the database. The data stored in the database can be created, retrieved, updated and deleted.

There are usually two types of DBMS: one is Relational RDBMS such as MySQL/PostgresSQL; the other is Non-relational (NRDBMS), such as MongoDB.

PART 5: Ace SQL Interviews

Interview Q2:

What are the joins in SQL? And what's their difference?

PART 5: Ace SQL Interviews

Interview Q2:

What are the joins in SQL? And what's their difference?

A JOIN Clause is used to combine rows from multiple tables, based on the related column between them.
There are **INNER, LEFT, RIGHT and FULL JOIN**.

INNER JOIN: Returns rows when records have matching values in both tables.
LEFT JOIN: Returns rows which are common between tables and all rows of left table.
FULL JOIN: Returns all rows from both sides.

PART 5: Ace SQL Interviews

Interview Q3:

What are the constraints?

PART 5: Ace SQL Interviews

Interview Q3:

What are the constraints?

A constraint is used to specify the limit on the data type of a table. It can be specified when creating or altering the table.

NOT NULL - Ensure null value cannot be stored.
UNIQUE - Values must be unique in the columns.
DEFAULT - If no value is specified, column has default value.
PRIMARY KEY - The column is the primary key of the table.

PART 5: Ace SQL Interviews

Interview Q4:

What is the difference between DELETE and DROP?

PART 5: Ace SQL Interviews

Interview Q4:

What is the difference between DELETE and DROP?

DELETE table removes the rows, however, the column attribute is intact. While **DROP** table removes a table and it cannot be rolled back from the database.

PART 5: Ace SQL Interviews

Interview Q5:

What’s the difference between UNION and UNION ALL?

PART 5: Ace SQL Interviews

Interview Q5:

What's the difference between UNION and UNION ALL?

UNION
UNION removes the duplicate records where all columns are the same.

UNION ALL
UNION ALL keeps all the records.

PART 5: Ace SQL Interviews

Interview Q6:

Write a SQL Query to find the names of employee name which starts with “B”?

PART 5: Ace SQL Interviews

Interview Q6:

Write a SQL Query to find the names of employee name which starts with “B”?

Analysis:

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

Operator	Condition	Example
LIKE	exact string comparison	col_name LIKE "ABC"
NOT LIKE	exact string inequality comparison	col_name NOT LIKE "ABCD"
%	match a sequence of zero or more characters (only with LIKE or NOT LIKE)	col_name LIKE "%AT%" (matches "AT", "AT&T", "CAT")
_	match a single character (only with LIKE or NOT LIKE)	col_name LIKE "AN_" (matches "AND", but not "AN")

Solution:

```
SELECT * FROM Employee
WHERE EmployeeName LIKE “B%”
```

PART 5: Ace SQL Interviews

Interview Q7:

Select the 2nd highest salary in an **Employee** table.

Id	Salary
1	100
2	200
3	300

SecondHighestSalary
200

PART 5: Ace SQL Interviews

Interview Q7:

Select the 2nd highest salary in an **Employee** table.

Analysis 1:

1. We first get the maximum salary by **MAX** function.
2. The second largest salary is the max salary after we exclude the maximum salary.
3. We can use a nested query to exclude the maximum salary.

Solution 1:

```
SELECT MAX(Salary) AS SecondHighestSalary
FROM Employee
WHERE Salary < (
        SELECT MAX(Salary) FROM Employee
)
```

PART 5: Ace SQL Interviews

Interview Q7:

Select the 2nd highest salary in an **Employee** table.

Analysis 2:

1. We can order the salary in descending way by combination of **ORDER BY** and **DESC**.
2. We then use the LIMIT and OFFSET clauses to subset the data. **LIMIT 1 OFFSET 1** will return the 2nd row.

Solution 2:

```
SELECT (
    SELECT DISTINCT Salary FROM Employee
    ORDER BY Salary DESC
    LIMIT 1
    OFFSET 1) AS SecondHighestSalary
```

PART 5: Ace SQL Interviews

Interview Q8:

Suppose that a website contains two tables, **Customers** table and **Orders** table. Write a SQL query to find all customers who never order anything.

Table: `Customers`.

```
+----+-------+
| Id | Name  |
+----+-------+
| 1  | Joe   |
| 2  | Henry |
| 3  | Sam   |
| 4  | Max   |
+----+-------+
```

Table: `Orders`.

```
+----+------------+
| Id | CustomerId |
+----+------------+
| 1  | 3          |
| 2  | 1          |
+----+------------+
```

Using the above tables as example, return the following:

```
+-----------+
| Customers |
+-----------+
| Henry     |
| Max       |
+-----------+
```

PART 5: Ace SQL Interviews

Interview Q8:

Suppose that a website contains two tables, **Customers** table and **Orders** table. Write a SQL query to find all customers who never order anything.

Analysis

1. We can get the customers who ordered by **INNER JOIN** Customers table and Order table
2. Output customers whose Id are not listed by **NOT IN**
3. Use a nested query to get the names

Solution

```
SELECT
Customers.Name AS "Customers"
FROM
Customers
WHERE
Customers.Id NOT IN (
    SELECT
    Customers.Id
    FROM Customers
    INNER JOIN Orders
    ON Orders.CustomerId = Customers.Id)
```

PART 5: Ace SQL Interviews

Interview Q9:

Write a SQL query to **delete** all duplicate email entries in a table named **Person**, keeping only unique emails based on its smallest **Id**.

```
+----+------------------+
| Id | Email            |
+----+------------------+
| 1  | john@example.com |
| 2  | bob@example.com  |
| 3  | john@example.com |
+----+------------------+
Id is the primary key column for this table.
```

```
+----+------------------+
| Id | Email            |
+----+------------------+
| 1  | john@example.com |
| 2  | bob@example.com  |
+----+------------------+
```

PART 5: Ace SQL Interviews

Interview Q9:

Write a SQL query to **delete** all duplicate email entries in a table named **Person**, keeping only unique emails based on its smallest **Id**.

Analysis

1. For duplicate email entries, they have multiple Id.
2. We should only keep **MIN** of the Id after **GROUP BY** at email level.
3. To exclude the other records, we can then use **NOT IN**.

Solution

```
DELETE FROM Person WHERE Id NOT IN
(SELECT * FROM (
    SELECT MIN(Id) FROM Person
    GROUP BY Email) AS p);
```

PART 5: Ace SQL Interviews

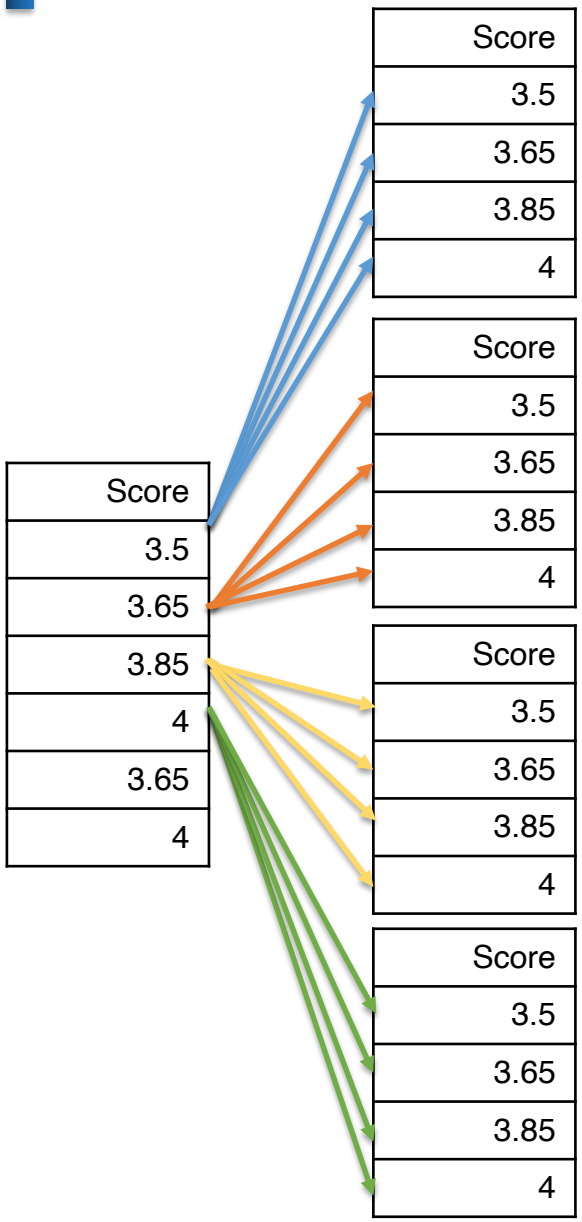
Interview Q10:

Write a SQL query to rank scores. If there is a tie between 2 scores, both should have same ranking. After a tie, the next ranking number should be next consecutive integer value.

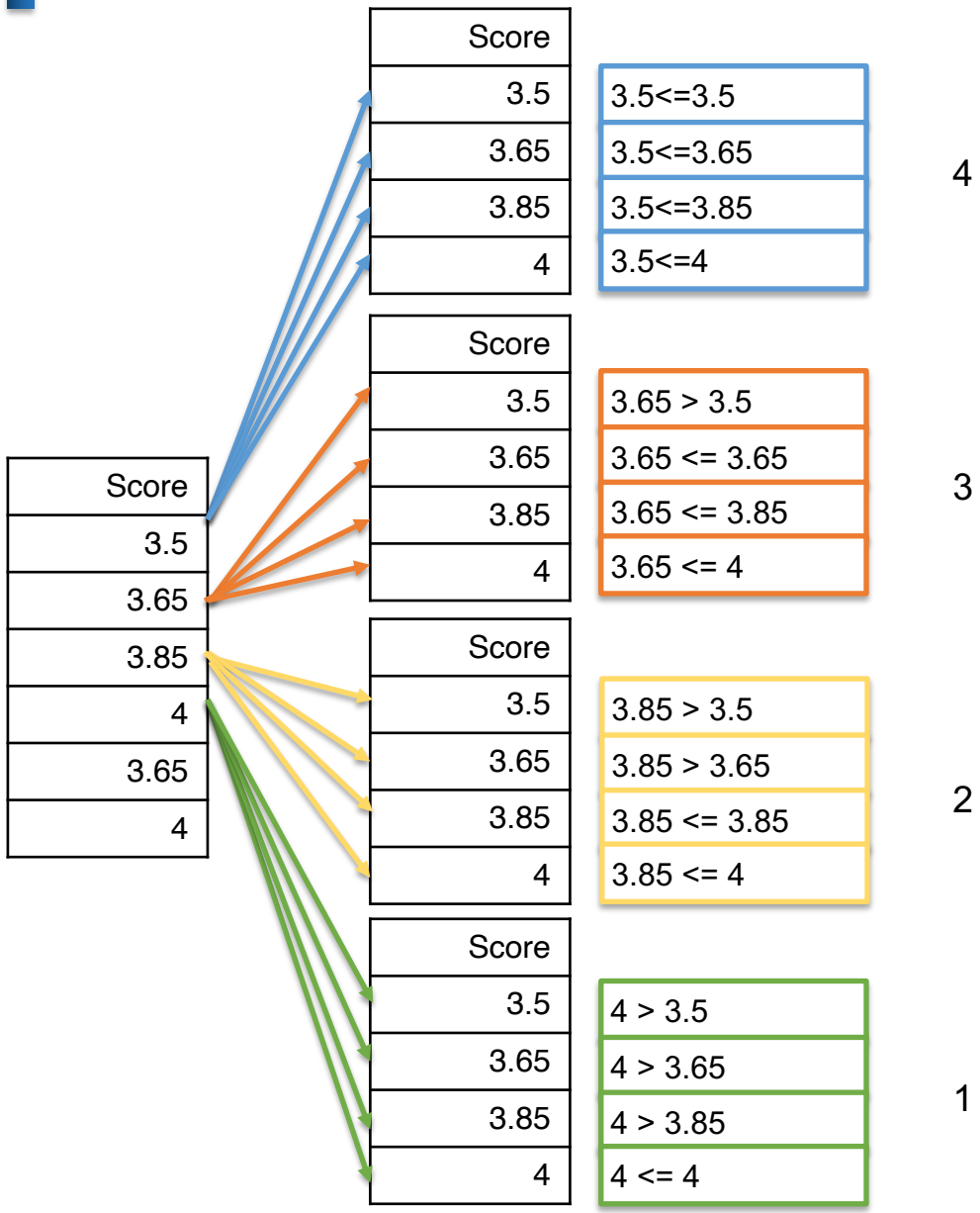
Id	Score
1	3.50
2	3.65
3	4.00
4	3.85
5	4.00
6	3.65

Score	Rank
4.00	1
4.00	1
3.85	2
3.65	3
3.65	3
3.50	4

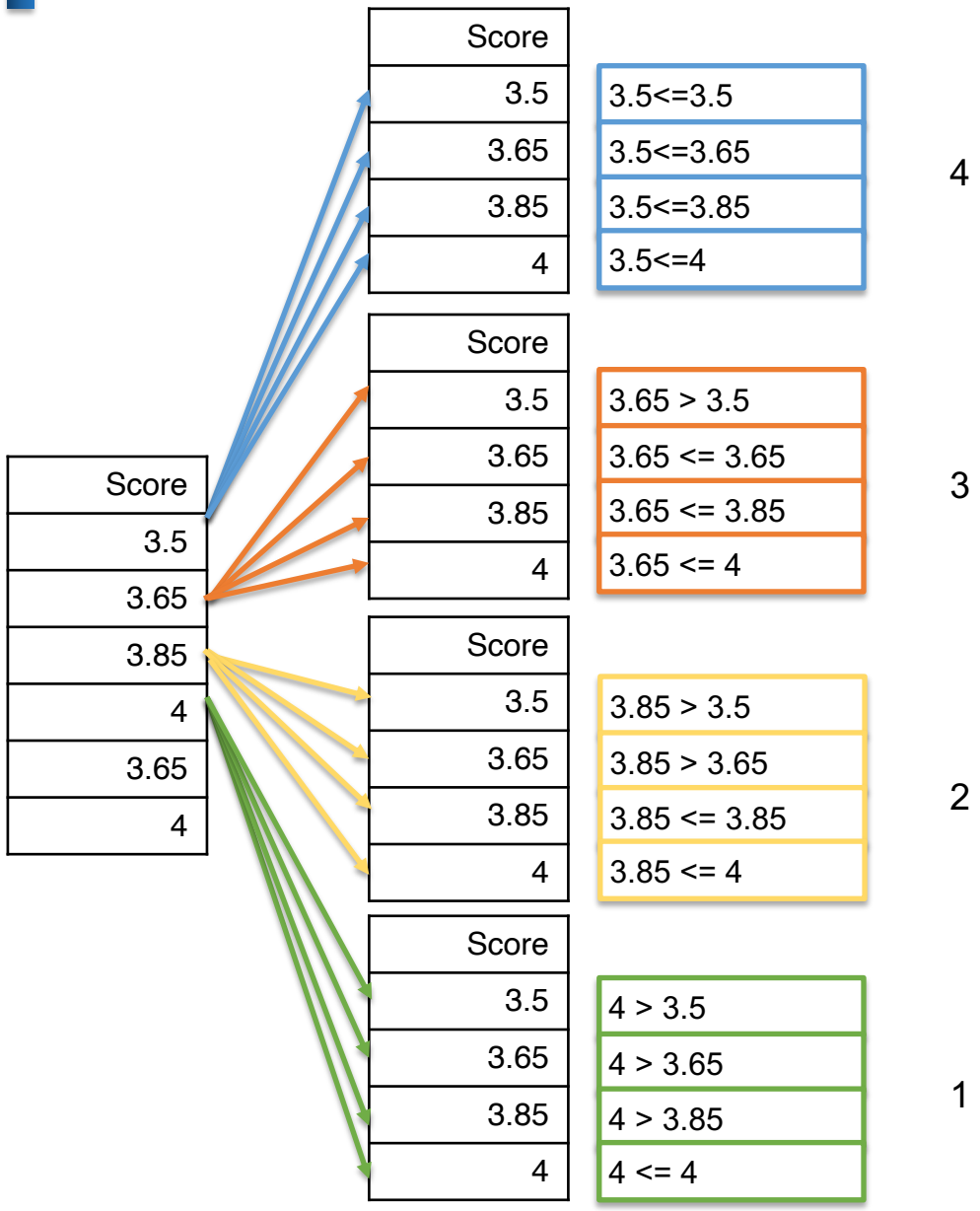
PART 5: Ace SQL Interviews



PART 5: Ace SQL Interviews



PART 5: Ace SQL Interviews



Original Table

Scores Table

Id	Score
1	3.5
2	3.65
3	4
4	3.85
5	4
6	3.65

List of Score

Score
3.5
3.65
4
3.85

```
SELECT DISTINCT Score FROM Scores;
```

PART 5: Ace SQL Interviews

Original Table

Scores Table

Id	Score
1	3.5
2	3.65
3	4
4	3.85
5	4
6	3.65

List of Score

Score
3.5
3.65
4
3.85

s.Id	s.Score	t.Score
1	3.5	3.5
1	3.5	3.65
1	3.5	4
1	3.5	3.85
2	3.65	3.5
2	3.65	3.65
2	3.65	4
2	3.65	3.85
3	4	3.5
3	4	3.65
3	4	4
3	4	3.85
4	3.85	3.5
4	3.85	3.65
4	3.85	4
4	3.85	3.85
5	4	3.5
5	4	3.65
5	4	4
5	4	3.85
6	3.65	3.5
6	3.65	3.65
6	3.65	4
6	3.65	3.85

PART 5: Ace SQL Interviews

Original Table

Scores Table

Id	Score
1	3.5
2	3.65
3	4
4	3.85
5	4
6	3.65

List of Score

Score
3.5
3.65
4
3.85

Permutation

```
SELECT s.Id, s.Score, t.Score FROM
    (SELECT DISTINCT Score FROM Scores) AS t,
    Scores AS s
```

s.Id	s.Score	t.Score
1	3.5	3.5
1	3.5	3.65
1	3.5	4
1	3.5	3.85
2	3.65	3.5
2	3.65	3.65
2	3.65	4
2	3.65	3.85
3	4	3.5
3	4	3.65
3	4	4
3	4	3.85
4	3.85	3.5
4	3.85	3.65
4	3.85	4
4	3.85	3.85
5	4	3.5
5	4	3.65
5	4	4
5	4	3.85
6	3.65	3.5
6	3.65	3.65
6	3.65	4
6	3.65	3.85

PART 5: Ace SQL Interviews

Original Table

Scores Table

Id	Score
1	3.5
2	3.65
3	4
4	3.85
5	4
6	3.65

List of Score

Score
3.5
3.65
4
3.85

```
SELECT s.Id, s.Score, t.Score FROM
    (SELECT DISTINCT Score FROM Scores) AS t,
    Scores AS s
```

s.Id	s.Score	t.Score
1	3.5	3.5
1	3.5	3.65
1	3.5	4
1	3.5	3.85
2	3.65	3.5
2	3.65	3.65
2	3.65	4
2	3.65	3.85
3	4	3.5
3	4	3.65
3	4	4
3	4	3.85
4	3.85	3.5
4	3.85	3.65
4	3.85	4
4	3.85	3.85
5	4	3.5
5	4	3.65
5	4	4
5	4	3.85
6	3.65	3.5
6	3.65	3.65
6	3.65	4
6	3.65	3.85

```
SELECT s.Score, t.Score FROM
    (SELECT DISTINCT Score FROM Scores) AS t,
    Scores AS s
WHERE s.Score <= t.Score
```

s.Id	s.Score	t.Score
1	3.5	3.5
1	3.5	3.65
1	3.5	4
1	3.5	3.85
2	3.65	3.65
2	3.65	4
2	3.65	3.85
3	4	4
4	3.85	4
4	3.85	3.85
5	4	4
6	3.65	3.65
6	3.65	4
6	3.65	3.85

PART 5: Ace SQL Interviews

```
SELECT s.Score, t.Score FROM
    (SELECT DISTINCT Score FROM Scores) AS t,
    Scores AS s
WHERE s.Score <= t.Score
```

s.Id	s.Score	t.Score
1	3.5	3.5
1	3.5	3.65
1	3.5	4
1	3.5	3.85
2	3.65	3.65
2	3.65	4
2	3.65	3.85
3	4	4
4	3.85	4
4	3.85	3.85
5	4	4
6	3.65	3.65
6	3.65	4
6	3.65	3.85

```
SELECT s.Score, COUNT(t.Score) AS "Rank" FROM
            (SELECT DISTINCT Score FROM Scores) AS t,
            Scores AS s
WHERE s.Score <= t.Score
GROUP BY s.Id, s.Score
```

s.Score	COUNT(t.Score) AS “Rank”
3.5	4
3.65	3
4	1
3.85	2
4	1
3.65	4

PART 5: Ace SQL Interviews

Interview Q10:

Write a SQL query to rank scores. If there is a tie between 2 scores, both should have same ranking. After a tie, the next ranking number should be next consecutive integer value.

Analysis

1. Rank is the number of scores in the Scores table without duplicates that is larger than or equal to the score. For example, there are six scores: 3.50, 3.65, 4.00, 3.85, 4.00, and 3.65. The distinct scores are 3.50, 3.65, 3.85 and 4.00. For score 4.00, only 4.00>=4.00, so its rank is 1. Similarly, 4.00>=3.65, 3.85>=3.65, 3.65>=3.65. 3.65 is ranked 3rd.
2. We need to get distinct scores.
3. Next, we can join the distinct score result tables and original Scores table. Filter scores that is larger than or equal to each score.
4. For each score in Scores table, count how many scores are larger than or equal to it.
5. Sort by descending order

Solution

```
SELECT s.Score, COUNT(t.Score) AS "Rank" FROM
        (SELECT DISTINCT Score FROM Scores) AS t,
        Scores AS s
WHERE s.Score <= t.Score
GROUP BY s.Id, s.Score
ORDER BY s.Score DESC;
```

PART 5: Ace SQL Interviews

QUERYING DATA FROM A TABLE

```
SELECT c1, c2 FROM t;
```
Query data in columns c1, c2 from a table

```
SELECT * FROM t;
```
Query all rows and columns from a table

```
SELECT c1, c2 FROM t
WHERE condition;
```
Query data and filter rows with a condition

```
SELECT DISTINCT c1 FROM t
WHERE condition;
```
Query distinct rows from a table

```
SELECT c1, c2 FROM t
ORDER BY c1 ASC [DESC];
```
Sort the result set in ascending or descending order

```
SELECT c1, c2 FROM t
ORDER BY c1
LIMIT n OFFSET offset;
```
Skip *offset* of rows and return the next n rows

```
SELECT c1, aggregate(c2)
FROM t
GROUP BY c1;
```
Group rows using an aggregate function

```
SELECT c1, aggregate(c2)
FROM t
GROUP BY c1
HAVING condition;
```
Filter groups using HAVING clause

QUERYING FROM MULTIPLE TABLES

```
SELECT c1, c2
FROM t1
INNER JOIN t2 ON condition;
```
Inner join t1 and t2

```
SELECT c1, c2
FROM t1
LEFT JOIN t2 ON condition;
```
Left join t1 and t1

```
SELECT c1, c2
FROM t1
RIGHT JOIN t2 ON condition;
```
Right join t1 and t2

```
SELECT c1, c2
FROM t1
FULL OUTER JOIN t2 ON condition;
```
Perform full outer join

```
SELECT c1, c2
FROM t1
CROSS JOIN t2;
```
Produce a Cartesian product of rows in tables

```
SELECT c1, c2
FROM t1, t2;
```
Another way to perform cross join

```
SELECT c1, c2
FROM t1 A
INNER JOIN t2 B ON condition;
```
Join t1 to itself using INNER JOIN clause

USING SQL OPERATORS

```
SELECT c1, c2 FROM t1
UNION [ALL]
SELECT c1, c2 FROM t2;
```
Combine rows from two queries

```
SELECT c1, c2 FROM t1
INTERSECT
SELECT c1, c2 FROM t2;
```
Return the intersection of two queries

```
SELECT c1, c2 FROM t1
MINUS
SELECT c1, c2 FROM t2;
```
Subtract a result set from another result set

```
SELECT c1, c2 FROM t1
WHERE c1 [NOT] LIKE pattern;
```
Query rows using pattern matching %, _

```
SELECT c1, c2 FROM t
WHERE c1 [NOT] IN value_list;
```
Query rows in a list

```
SELECT c1, c2 FROM t
WHERE  c1 BETWEEN low AND high;
```
Query rows between two values

```
SELECT c1, c2 FROM t
WHERE  c1 IS [NOT] NULL;
```
Check if values in a table is NULL or not

PART 5: Ace SQL Interviews

MANAGING TABLES

```
CREATE TABLE t (
    id INT PRIMARY KEY,
    name VARCHAR NOT NULL,
    price INT DEFAULT 0
);
```
Create a new table with three columns

```
DROP TABLE t ;
```
Delete the table from the database

```
ALTER TABLE t ADD column;
```
Add a new column to the table

```
ALTER TABLE t DROP COLUMN c ;
```
Drop column c from the table

```
ALTER TABLE t ADD constraint;
```
Add a constraint

```
ALTER TABLE t DROP constraint;
```
Drop a constraint

```
ALTER TABLE t1 RENAME TO t2;
```
Rename a table from t1 to t2

```
ALTER TABLE t1 RENAME c1 TO c2 ;
```
Rename column c1 to c2

```
TRUNCATE TABLE t;
```
Remove all data in a table

USING SQL CONSTRAINTS

```
CREATE TABLE t(
   c1 INT, c2 INT, c3 VARCHAR,
   PRIMARY KEY (c1,c2)
);
```
Set c1 and c2 as a primary key

```
CREATE TABLE t1(
   c1 INT PRIMARY KEY,
   c2 INT,
   FOREIGN KEY (c2) REFERENCES t2(c2)
);
```
Set c2 column as a foreign key

```
CREATE TABLE t(
   c1 INT, c1 INT,
   UNIQUE(c2,c3)
);
```
Make the values in c1 and c2 unique

```
CREATE TABLE t(
  c1 INT, c2 INT,
  CHECK(c1> 0 AND c1 >= c2)
);
```
Ensure c1 > 0 and values in c1 >= c2

```
CREATE TABLE t(
    c1 INT PRIMARY KEY,
    c2 VARCHAR NOT NULL
);
```
Set values in c2 column not NULL

MODIFYING DATA

```
INSERT INTO t(column_list)
VALUES(value_list);
```
Insert one row into a table

```
INSERT INTO t(column_list)
VALUES (value_list),
       (value_list), ....;
```
Insert multiple rows into a table

```
INSERT INTO t1(column_list)
SELECT column_list
FROM t2;
```
Insert rows from t2 into t1

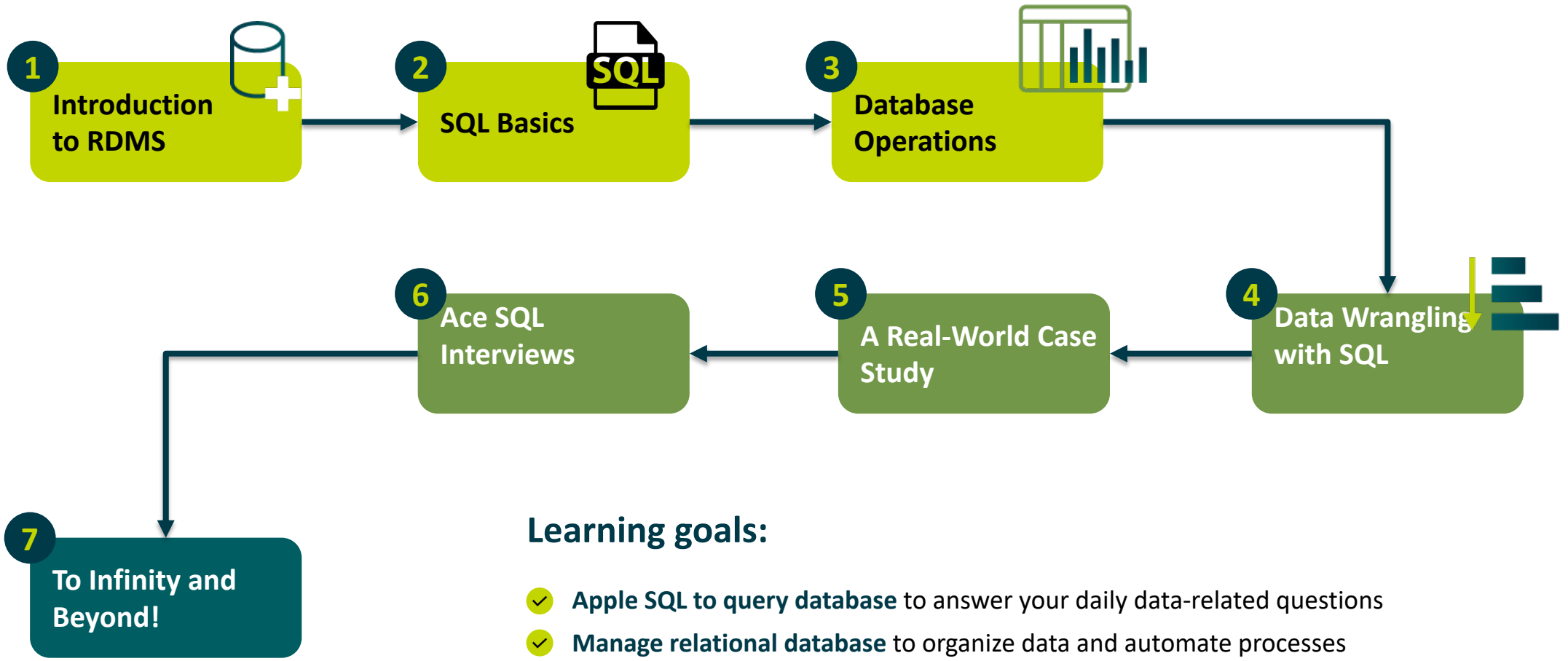
```
UPDATE t
SET c1 = new_value;
```
Update new value in the column c1 for all rows

```
UPDATE t
SET c1 = new_value,
    c2 = new_value
WHERE condition;
```
Update values in the column c1, c2 that match the condition

```
DELETE FROM t;
```
Delete all data in a table

```
DELETE FROM t
WHERE condition;
```
Delete subset of rows in a table

COURSE AGENDA



Learning goals:

- **Apple SQL to query database** to answer your daily data-related questions
- **Manage relational database** to organize data and automate processes
- **Be Comfortable** in SQL technical interviews