1. DATA BASE CLASS

What is database?

A database is a collection of information that is organized so that it can easily be accessed, managed, and updated.

What is RDBMS?

RDBMS stands for Relational Database Management System. RDBMS data is structured in database tables, fields and records. Each RDBMS table consists of database table rows. Each database table row consists of one or more database table fields.

RDBMS store the data into collection of tables, which might be related by common fields (database table columns). RDBMS also provide relational operators to manipulate the data stored into the database tables. Most RDBMS use SQL as database query language.

2. SQL

DDL- Data Definition Language
DML- Data Manipulation Language
DCL- Data Control Language
TCL- Transaction Control Language

2.1 DDL

Data Definition Language (DDL) statements are used to define the database structure or schema. Some examples:

- CREATE : To create objects in the database
- ALTER: Alters the structure of the database
- DROP:Delete objects from the database
- TRUNCATE : Remove all records from a table, including all spaces
    - allocated for the records are removed
- COMMENT : Add comments to the data dictionary
- RENAME : Rename an object

2.2 DML

Data Manipulation Language (DML) statements are used for managing data within schema objects. Some examples:

1. **SELECT - retrieve data from the a database**

```
SELECT column_name,column_name
FROM table_name
```

```
SELECT *
FROM table_name
```

2. **SELECT DISTINCT Syntax**

```
SELECT DISTINCT column_name,column_name
FROM table_name;
```

3. **INSERT - insert data into a table**

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

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

4. **UPDATE - updates existing data within a table**

```
UPDATE table name
SET column1=value1,column2=value2,...
WHERE some column=some value;
```

5. **DELETE - deletes all records from a table, the space for the records remain**

```
DELETE FROM table_name
WHERE some_column=some_value;
```

6. **WHERE Clause - The WHERE clause is used to extract only those records that fulfill a specified criterion.**

```
SELECT column_name,column_name
FROM table_name
WHERE column_name operator value;
```

7. **AND Operator - The following SQL statement selects all customers from the country "Germany" AND the city "Berlin", in the "Customers" table:**

```
SELECT * FROM Customers
WHERE Country='Germany'
AND City='Berlin';
```

8. **OR Operator - The following SQL statement selects all customers from the city "Berlin" OR "München", in the "Customers" table:**

```
SELECT * FROM Customers
WHERE City='Berlin'
OR City='München';
```

9. Combining AND & OR

You can also combine AND and OR (use parenthesis to form complex expressions).
The following SQL statement selects all customers from the country "Germany" AND the city must be equal to "Berlin" OR "München", in the "Customers" table:

```
SELECT * FROM Customers
WHERE Country='Germany
AND (City='Berlin' OR City='München')
```

10.ORDER BY Keyword

The ORDER BY keyword is used to sort the result-set by one or more columns.
The ORDER BY keyword sorts the records in ascending order by default. To sort the records in a descending order, you can use the DESC keyword.

```
SELECT column_name, column_name FROM table_name
ORDER BY column_name,column_name ASC|DESC
```

11.LIKE Operator

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

```
SELECT column_name(s)
FROM table_name
WHERE column_name LIKE pattern;
```

12.Wildcard Characters

In SQL, wildcard characters are used with the SQL LIKE operator.
SQL wildcards are used to search for data within a table.
With SQL, the wildcards are:

Wildcard	Description
%	A substitute for zero or more characters
_	A substitute for a single character
[*charlist*]	Sets and ranges of characters to match
[^*charlist*] or [!*charlist*]	Matches only a character NOT specified within the brackets

Using the SQL % Wildcard

Example 1

The following SQL statement selects all customers with a City starting with "ber":

```
SELECT * FROM Customers
WHERE City LIKE 'ber%';
```

Example 2

The following SQL statement selects all customers with a City containing the pattern "es":

```
SELECT * FROM Customers
WHERE City LIKE '%es%';
```

Example 3

The following SQL statement selects all customers with a City starting with any character, followed by "erlin":

```
SELECT * FROM Customers
WHERE City LIKE '_erlin';
```

Example 4

The following SQL statement selects all customers with a City starting with "L", followed by any character, followed by "n", followed by any character, followed by "on":

```
SELECT * FROM Customers
WHERE City LIKE 'L_n_on';
```

Example 5

Using the SQL [charlist] Wildcard

The following SQL statement selects all customers with a City starting with "b", "s", or "p":

```
SELECT * FROM Customers
WHERE City LIKE '[bsp]%';
```

Example 6

The following SQL statement selects all customers with a City starting with "a", "b", or "c":

```
SELECT * FROM Customers
WHERE City LIKE '[a-c]%';
```

Example 7

The following SQL statement selects all customers with a City NOT starting with "b", "s", or "p":

```
SELECT * FROM Customers
WHERE City LIKE '[!bsp]%';
```

13.SQL Aliases

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

Basically aliases are created to make column names more readable.

SQL Alias Syntax for Columns

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

SQL Alias Syntax for Tables

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

14.BETWEEN Operator

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

SQL BETWEEN Syntax

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

2.3 DCL

Data Control Language (DCL) statements. Some examples:

1. GRANT : Gives user's access privileges to database
2. REVOKE : Withdraw access privileges given with the GRANT command

2.4 TCL

Transaction Control (TCL) statements are used to manage the changes made by DML statements. It allows statements to be grouped together into logical transactions.

1. COMMIT : Save work done
2. SAVEPOINT : Identify a point in a transaction to which you can later roll back
3. ROLLBACK : Restore database to original since the last COMMIT
4. SET TRANSACTION: Change transaction options like isolation level and what rollback segment to use

2.5 SQL Advanced

SQL PRIMARY KEY Constraint

The PRIMARY KEY constraint uniquely identifies each record in a database table.

Primary keys must contain unique values.

A primary key column cannot contain NULL values.

Each table should have a primary key, and each table can have only ONE primary key.

SQL FOREIGN KEY Constraint

A FOREIGN KEY in one table points to a PRIMARY KEY in another table.

Let's illustrate the foreign key with an example. Look at the following two tables:

The "Persons" table:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

The "Orders" table:

O_Id	OrderNo	P_Id
1	77895	3
2	44678	3
3	22456	2
4	24562	1

SQL JOINS

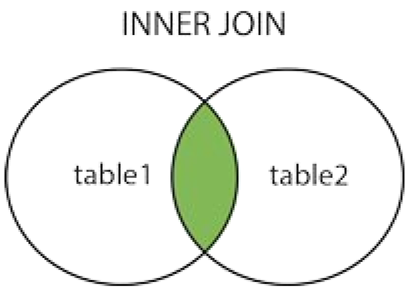
SQL joins are used to combine rows from two or more tables.

1. **INNER JOIN**

The INNER JOIN keyword selects all rows from both tables as long as there is a match between the columns in both tables.

SELECT *column_name(s)* FROM *table1* INNER JOIN *table2* ON *table1.column_name=table2.column_name*;
SELECT *column_name(s)* FROM *table1* JOIN *table2* ON *table1.column_name=table2.column_name*;

PS! INNER JOIN is the same as JOIN.



2. **LEFT JOIN Keyword**

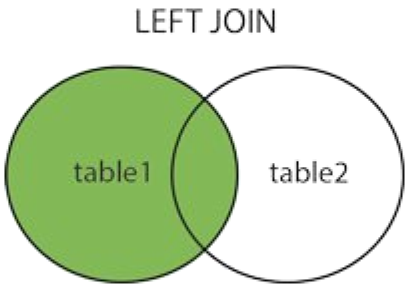
The LEFT JOIN keyword returns all rows from the left table (table1), with the matching rows in the right table (table2). The result is NULL in the right side when there is no match.

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

or:

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

PS! In some databases LEFT JOIN is called LEFT OUTER JOIN.



3. **RIGHT JOIN Keyword**

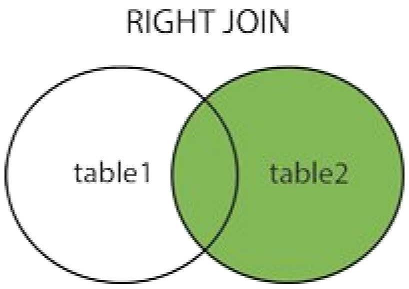
The RIGHT JOIN keyword returns all rows from the right table (table2), with the matching rows in the left table (table1). The result is NULL in the left side when there is no match.

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

or:

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

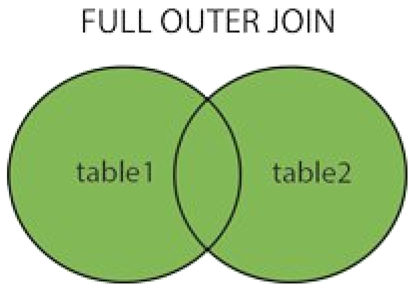
PS! In some databases RIGHT JOIN is called RIGHT OUTER JOIN.



4. **FULL OUTER JOIN Keyword**

The FULL OUTER JOIN keyword returns all rows from the left table (table1) and from the right table (table2).
The FULL OUTER JOIN keyword combines the result of both LEFT and RIGHT joins.

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


FULL OUTER JOIN
table1
table2