1.	**Java features**	• Object oriented • Platform Independent • Simple, Secure • Portable • Multi-tasking
2.	**Object Oriented Programming**	It is used to structure a software program into simple, reusable pieces of code blueprints (usually called classes), which are used to create individual instances of objects.
3.	**object**	Any entity that has state and behaviour is known as an object. An Object can be defined as an instance of a class
4.	**Class**	Collection of *objects* is called class. It is a logical entity. A class can also be defined as a blueprint from which you can create an individual object.
5.	**Wrapper class**	A **Wrapper class** is a **class** whose object wraps or contains primitive data types. ... In other words, we can wrap a primitive value into a **wrapper class** object. Need of **Wrapper Classes**. They convert primitive data types into objects.
6.	**Inheritance**	When one object acquires all the properties and behaviours of a parent object, it is known as inheritance. In inheritance, sub class inherits all the properties but there is one exception. The constructor of the super class are never inherited by the sub class. It provides code reusability. It is used to achieve runtime polymorphism.
7.	**Polymorphism**	If one task is performed in different ways, it is known as polymorphism. In Java, we use method overloading and method overriding to achieve polymorphism.
8.	**Abstraction**	Hiding internal details and showing functionality is known as abstraction. In Java, we use abstract class and interface to achieve abstraction.
9.	**JVM**	JVM (Java Virtual Machine) is an abstract machine. It is called a virtual machine because it doesn't physically exist. It is a specification that provides a runtime environment in which Java byte code can be executed. It can also run those programs which are written in other languages and compiled to Java byte code.
10.	**JRE**	JRE is an acronym for Java Runtime Environment. It is also written as Java RTE. The Java Runtime Environment is a set of software tools which are used for developing Java applications. It is used to provide the runtime environment. It is the implementation of JVM. It physically exists. It contains a set of libraries + other files that JVM uses at runtime.
11.	**JDK**	JDK is an acronym for Java Development Kit. The Java Development Kit (JDK) is a software development environment which is used to develop Java applications and applets. It physically exists. It contains JRE + development tools.
12.	**Why secure**	**Java programs run inside a virtual machine** which is known as a sandbox. Java does not support explicit pointer. Byte-code verifier

		checks the code fragments for illegal code that can violate access right to object.
13.	**Why pointers not used in java**	Java has a **robust security model** and disallows pointer arithmetic for the same reason. ... No pointer support make Java more secure because they point to memory location or used for memory management that loses the security as we use them directly.
14.	**Partially object oriented and not 100%**	As Java supports usual declaration of data variables, it is partial implementation of OOP. Because according to rules of OOP, object constructors must be used, even for declaration of variables.
15.	**Java Platform independent**	The meaning of platform-independent is that the java compiled code (byte code) can run on all operating systems. Java has WORA concept. (Write Once Run Anywhere).
16.	**Constructor**	In Java, a constructor is a block of codes similar to the method. It is called when an instance of the class is created. ... It is a special type of method which is used to initialize the object. Every time an object is created using the new () keyword, at least one constructor is called.
17.	**Multi-Threading**	Multithreading refers to two or more **threads** executing concurrently in a single program.
18.	**Thread**	**Thread is the light weight process. Threads** allows a program to operate more efficiently by doing multiple things at the same time. **Threads** can be **used** to perform complicated tasks in the background without interrupting the main program.
19.	**Exception**	In Java "**an event that occurs during the execution of a program that disrupts the normal flow of instructions"** is called an exception. This is generally an unexpected or unwanted event which can occur either at compile-time or run-time in application code.
20.	**5 exception key words**	Try, **catch**, throw, throws, and finally.
21.	**Exception Handling**	The **Exception Handling in Java** is one of the powerful *mechanisms to handle* the runtime errors so that the normal flow of the application can be maintained.
22.	**Throw and Throws**	Java throw keyword is used to explicitly Throw an exception. / Java throws keyword is used To declare an exception. Throw is followed by an instance. / Throws is followed by class. You cannot throw multiple exceptions. / You can declare multiple exceptions
23.	**Final and Finally**	• The **final** keyword can be used with class method and variable. A final class cannot be instantiated, a final method cannot be overridden and a final variable cannot be reassigned. • The **finally** keyword is used to create a block of code that follows a try block. A finally block of code always executes, whether or not an exception has occurred.
24.	**HTML**	HTML (Hypertext Mark-up Language) is the code that is used to structure a web page and its content. For example, content could be structured within a set of paragraphs, a list of bulleted points, or using images and data tables.
25.	**CSS**	CSS (Cascading Style sheets) is the language for describing the presentation of Web pages, including colours, layout, and fonts. It allows one to adapt the presentation to different types of devices, such as large screens, small screens, or printers.

26.	**XML**	XML stands for extensible Mark-up Language · XML is a mark-up language much like HTML. XML provides a standard method to access information, making it easier for applications and devices of all kinds to use, store, transmit, and display data.
27.	**Collections**	Collection is a group of objects. The Collection in Java is a framework that provides an architecture to store and manipulate the group of objects. Java Collections can achieve all the operations that you perform on a data such as searching, sorting, insertion, manipulation, and deletion.
28.	**Diff between String buffer and String Builder**	String Buffer is *synchronized* i.e. thread safe. It means two threads can't call the methods of String Buffer simultaneously. / String Builder is *non-synchronized* i.e. not thread safe. It means two threads can call the methods of String Builder simultaneously. String Buffer is *less efficient* than String Builder. / String Builder is *more efficient* than String Buffer.
29.	**Jump Iterator**	basically this jump iterator is trying to iterator each element with one distance, for example, for 1,2,3,4,5, it will return 1,3,5.
30.	**For and For each loop**	The for loop is best **for iterating over name-value** pairs, and the for Each is loop best for iterating over values, for example arrays or objects.
31.	**Abstract class**	Abstract class is **a restricted class that cannot be used to create objects** (to access it, it must be inherited from another class).
32.	**Abstract Method**	Abstract method: can only be used in an abstract class, and it does not have a body. The body is provided by the subclass (inherited from).
33.	**Overloading**	Overloading occurs **when two or more methods in one class have the same method name but different parameters**.
34.	**Overriding**	Overriding occurs when two methods have the same method name and parameters. One of the methods is in the parent class, and the other is in the child class.
35.	**Data Type**	Data type **specifies the size and type of values that can be stored** in an identifier. ... Data types in Java are classified into two types: Primitive—which include Integer, Character, Boolean, and Floating Point. Non-primitive—which include Classes, Interfaces, and Arrays.
36.	**C vs Java difference**	C is a **middle-level** language. / Java is a **high-level** language. C is a structural and **procedure-oriented** programming language. / Java is an **object-orient** language It follows the **top-down** approach to design the application. / It follows the **bottom-up** approa application. It is a **compiled** language. / It is an **interpreted** language.
37.	**C++ vs Java difference**	C++ is platform-dependent. / Java is platform-independent. C++ supports the goto statement. / Java doesn't support the goto statement C++ supports operator overloading. / Java doesn't support operator overloadi C++ supports pointer, structure and union / Java doesn't support those

38.	**Finalize()**	Finalize is a method which is used in the garbage collections. This method will be called before deleting the object.
39.	**Garbage Collections**	In java, garbage means unreferenced objects. Garbage Collection is process of reclaiming the runtime unused memory automatically. It makes java **memory efficient** because garbage collector removes the unreferenced objects from heap memory.
40.	**Abstract method**	The methods declared inside the abstract class are abstract methods. Abstract methods have only declaration there is no definition. Abstract methods should be overridden in the sub classes.
41.	**Interface**	Interface is a pure abstract class. Interface allows us to declare only abstract methods. Doesn't allows to declare concrete methods. Methods inside the interface is by default public abstract and the variable inside the interface is by default pubic final. Ex: Debit card.
42.	**Keywords**	Keywords in java are a special characters which is used to perform some specific tasks. Like final, static, abstract, interface etc…
43.		Class to class → extends Class to interface → implements Interface to Interface → extends
44.	**Package**	Package in java is like a folder. It is like file directory concepts. Packages won't allow us to create the same class name as we used it before. Mainly package is used for to avoid the namespace collision.
45.	**Types of exceptions**	Checked exception (happens in the compile time) • IOException • SQL Exception • Interrupted exception Unchecked Exception (happens in the run time) • IndexOutofBound Exception • NullpointerException • Arithmetic Exception
46.	**Auto boxing**	The process of converting primitive data types in to a wrapper class in known as auto boxing.
47.	**Un boxing**	The process of converting wrapper class in to a primitive data types in known as Un boxing.
48.	**Type casting/conversion**	The process of converting one data types in to another data type explicitly is known as type casting. The compiler automatically changes the data type is known as type conversion.
49.	**Super**	The **super** keyword in Java is a reference variable which is used to refer immediate parent class object. We can call the super class constructor by using the super key word.
50.	**This**	This keyword can be used to refer current class instance variable. If there is ambiguity between the instance variables and parameters, this keyword resolves the problem of ambiguity. It can be used to call the overloaded constructor.
51.	**String methods**	• Length(); • isUpperCase(); • ToLowerCase();

		• SubString(); • charAt(); • IndexOf(); • Split(); • Equals();
52.	**JIT**	JIT is Just In Time Compiler. JIT is a part of the JVM. It compiles some part of the source code in to machine code. In increases the performance of execution.
53.	**API**	Java application programming interface (API) is **a list of all classes that are part** of the Java development kit (JDK). It includes all Java packages, classes, and interfaces, along with their methods, fields, and constructors.
54.	**Trim()**	Trim is used to remove the unwanted white spaces which are placed in the beginning and end of the string.
55.	**Console**	Console is a terminal where we can perform tasks by giving the commands.
56.	**Early Binding / Late binding**	Early binding is a compile time polymorphism and late binding is the run time polymorphism.
57.	**Inner class / outer class**	If one class is placed inside another class then it is known as nested class. In the nested class the class which holds the inner class is said to be an outer class.
58..	**Printstacktrace()**	The printStackTrace() method in Java is **a tool used to handle exceptions and errors**. It is a method of Java's throwable class which prints the throwable along with other details like the line number and class name where the exception occurred.
59.	**DataBase Model**	• Network model • Hierarchical model • Relational model
60.	**Types of java applications**	• Standalone application • Web application • Mobile application • Enterprise application
61.	**Collection methods**	• Size(); • isEmpty(); • contains(); • add(); • put(); • iterator(); • toArray();
62.	**View levels in data base**	• external level (individual user view) • conceptual level (community user view) • Internal level (storage view)
63.	**cardinality**	No of tuples
64.	**degree**	No of attributes
65.	**projection**	Column wise
66.	**selection**	Row wise
67.	**BLOB**	To add images in the DB

68.	**JDBC Drivers**	Bridge between the java program and the data base
69.	**4 types of drivers**	1. Type-1 JDBC bridge 2. Type-2 Native API driver 3. Type-3 network protocol driver 4. Type-4 protocol driver
70.	**5 major steps to connect the DB**	1. Forname() 2. getConnection(); 3. Statement 4. Resultset 5. While(rset.next())
71.	**3 types of statements**	1. Prepared statement 2. Callable statement 3. Normal statement
72.	**Enumeration**	The **Enum in Java** is a data type which contains a fixed set of constants. Enums are used to create our own data type like classes. It can be used for days of the week (SUNDAY, MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, and SATURDAY) , directions (NORTH, SOUTH, EAST, and WEST),
73.	**CSS Selectors**	• Element selector (p { }) • Id selector (#) • Class selector (.) • Universal selector (*) • Grouping selectors (p h1 h2)

74.	**element Selector**	Here, all <p> elements on the page will be center-aligned, with a red text color:
		
		p {
		text-align: center;
		color: red;
		}

75.	**id Selector**	The id selector uses the id attribute of an HTML element to select a specific element.
		
		The id of an element is unique within a page, so the id selector is used to select one unique element!
		
		To select an element with a specific id, write a hash (#) character, followed by the id of the element.
		
		Example
		
		The CSS rule below will be applied to the HTML element with id="para1":
		
		#para1 {
		text-align: center;
		color: red;
		}

76.	**CSS class Selector**	The class selector selects HTML elements with a specific class attribute.
		
		To select elements with a specific class, write a period (.) character, followed by the class name.
		
		Example
		
		In this example all HTML elements with class="center" will be red and center-aligned:
		
		`.center {`
		`  text-align: center;`
		`  color: red;`
		`}`

77.	**Universal Selector**	The universal selector (*) selects all HTML elements on the page.
		
		Example
		
		The CSS rule below will affect every HTML element on the page:
		
		`* {`
		`  text-align: center;`
		`  color: blue;`
		`}`

78.	**Grouping Selector**	The grouping selector selects all the HTML elements with the same style definitions.
		
		Look at the following CSS code (the h1, h2, and p elements have the same style definitions):
		
		`h1 {`
		`  text-align: center;`
		`  color: red;`
		`}`
		
		`h2 {`
		`  text-align: center;`
		`  color: red;`
		`}`
		
		`p {`
		`  text-align: center;`
		`  color: red;`
		`}`

79.	**Normalization**	○ Normalization is the process of organizing the data in the database. ○ Normalization is used to minimize the redundancy from a relation or set of relations. It is also used to eliminate the undesirable characteristics like Insertion, Update and Deletion Anomalies.

<table>
<tr><td></td><td></td><td>

Normalization divides the larger table into the smaller table and links them using relationship.
The normal form is used to reduce redundancy from the database table.

</td></tr>
<tr><td>80.</td><td>DE normalization</td><td>When we normalize tables, we break them into multiple smaller tables. So when we want to retrieve data from multiple tables, we need to perform some kind of join operation on them. In that case, we use the denormalization technique that eliminates the drawback of normalization.</td></tr>
<tr><td>81.</td><td>Types of normalizations</td><td>
<table>
<tr><td>1NF</td><td>A relation is in 1NF if it contains an atomic value.</td></tr>
<tr><td>2NF</td><td>A relation will be in 2NF if it is in 1NF and all non-key attributes
Are fully functional dependent on the primary key.</td></tr>
<tr><td>3NF</td><td>A relation will be in 3NF if it is in 2NF and no transition dependency exists.</td></tr>
<tr><td>4NF</td><td>A relation will be in 4NF if it is in Boyce Codd normal form and has no
Multi-valued dependency.</td></tr>
<tr><td>5NF</td><td>A relation is in 5NF if it is in 4NF and not contains any join dependency an
d joining should be lossless.</td></tr>
</table>
</td></tr>
<tr><td>82.</td><td>JDBC connection program & syntaxes</td><td>
<pre>
import java.sql.*;
class MysqlCon{
public static void main(String args[]){
try{
Class.forName("com.mysql.jdbc.Driver");
Connection con=DriverManager.getConnection(
"jdbc:mysql://localhost:3306/sonoo","root","root");
//here sonoo is database name, root is username and password
Statement stmt=con.createStatement();
. ResultSet rs=stmt.executeQuery("select * from emp");
. while(rs.next())
. System.out.println(rs.getInt(1)+" "+rs.getString(2)+" "+rs.getString(3));
. con.close();
. }catch(Exception e){ System.out.println(e);}
. }
. }
</pre>
</td></tr>
<tr><td>83.</td><td>DB schema</td><td>

The data which is stored in the database at a particular moment of time is called an instance of the database.
The overall design of a database is called schema.
A database schema is the skeleton structure of the database. It represents the logical view of the entire database.
A schema contains schema objects like table, foreign key, primary key, views, columns, data types, stored procedure, etc.

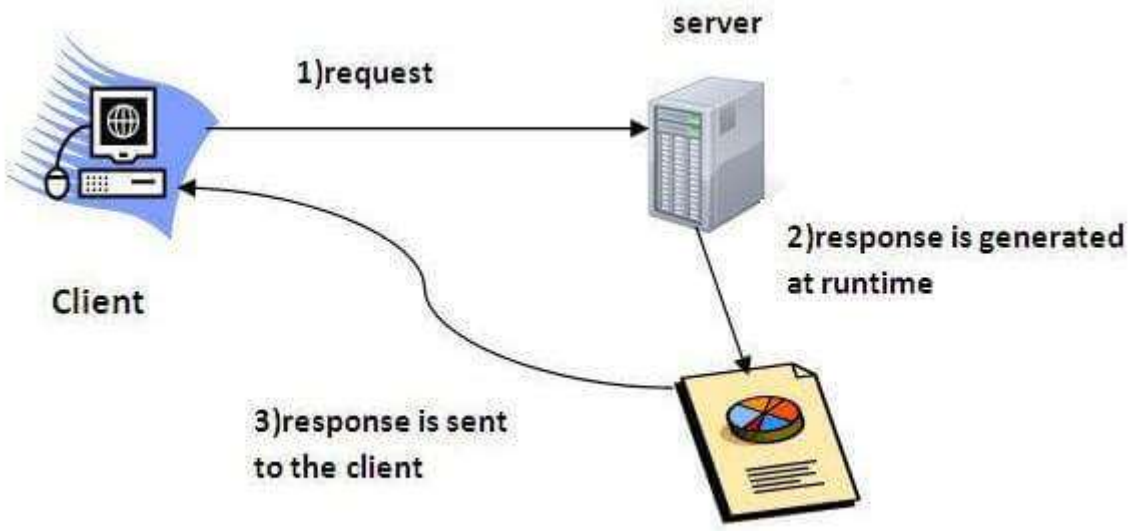
</td></tr>
<tr><td>84.</td><td>Servlets</td><td>

Servlet is a technology which is used to create a web application.

</td></tr>
</table>

<table>
<tr>
<td></td>
<td></td>
<td>

Servlet is an API that provides many interfaces and classes including documentation.
Servlet is an interface that must be implemented for creating any Servlet.
Servlet is a class that extends the capabilities of the servers and responds to the incoming requests. It can respond to any requests.
Servlet is a web component that is deployed on the server to create a dynamic web page.



</td>
</tr>
<tr>
<td>85.</td>
<td>Applets</td>
<td>
Applet is a special type of program that is embedded in the webpage to generate the dynamic content. It runs inside the browser and works at client side.
<h2>Advantage of Applet</h2>
There are many advantages of applet. They are as follows:

It works at client side so less response time.
Secured
It can be executed by browsers running under many plateforms, including Linux, Windows, Mac Os etc.

<h2>Drawback of Applet</h2>

Plugin is required at client browser to execute applet.

</td>
</tr>
<tr>
<td>86.</td>
<td>CREATE, DROP , BACKUP</td>
<td>

```
CREATE DATABASE databasename;

DROP DATABASE databasename;

BACKUP DATABASE databasename
TO DISK = 'filepath';

BACKUP DATABASE databasename
TO DISK = 'filepath'
WITH DIFFERENTIAL;
```

</td>
</tr>
<tr>
<td>87.</td>
<td>CREATE A TABLE</td>
<td>

```
CREATE TABLE table_name (
    column1 datatype,
    column2 datatype,
    column3 datatype,
   ....
);

CREATE TABLE Persons (
    PersonID int,
    LastName varchar(255),
    FirstName varchar(255),
    Address varchar(255),
```

</td>
</tr>
</table>

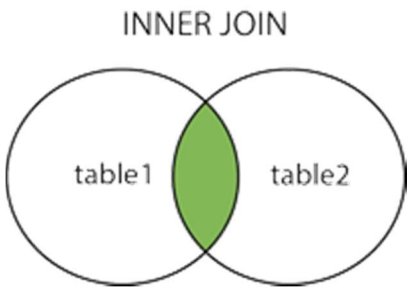
		City varchar(255)); ```
88.	**ALTER a table**	``` ALTER TABLE table_name ADD column_name datatype; ALTER TABLE Customers ADD Email varchar(255); ```

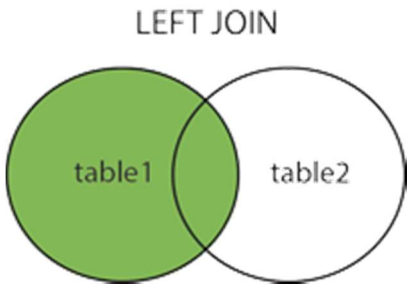
89.	**PRIMARY & FOREIGN keys**	```
		CREATE TABLE Orders (
		    OrderID int NOT NULL,
		    OrderNumber int NOT NULL,
		    PersonID int,
		    PRIMARY KEY (OrderID),
		    FOREIGN KEY (PersonID) REFERENCES Persons(PersonID)
		);
		``

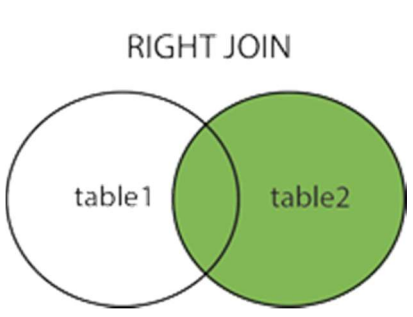
90.	**To select an entire table**	``` SELECT * FROM table_name; ```
91.	**DISTINCT**	The SELECT DISTINCT statement is used to return only distinct (different) values. ``` SELECT DISTINCT Country FROM Customers; ```
92.	**AND OR NOT**	``` SELECT * FROM Customers WHERE Country='Mexico'; SELECT * FROM Customers WHERE Country='Germany' AND City='Berlin'; SELECT * FROM Customers WHERE City='Berlin' OR City='München'; ``` The following SQL statement selects all fields from "Customers" where country is NOT "Germany": ``` SELECT * FROM Customers WHERE NOT Country='Germany'; ```
92.	**ORDER BYE**	``` SELECT * FROM Customers ORDER BY Country DESC; SELECT * FROM Customers ORDER BY Country ASC, CustomerName DESC; ```
93.	**INSERT**	``` INSERT INTO Customers (CustomerName, ContactName, Address, City, PostalCode, Country) VALUES ('Cardinal', 'Tom B. Erichsen', 'Skagen 21', 'Stavanger', '4006', 'Norway'); ```
94.	**Starts with “a”**	The following SQL statement selects all customers with a CustomerName starting with "a": ``` SELECT * FROM Customers WHERE CustomerName LIKE 'a%'; ```
95.	**UPDATE**	``` UPDATE table_name SET column1 = value1, column2 = value2, ... WHERE condition; UPDATE Customers SET ContactName = 'Alfred Schmidt', City= 'Frankfurt' WHERE CustomerID = 1; ```

96.	**DELETE**	`DELETE FROM Customers WHERE CustomerName='Alfreds Futterkiste';`

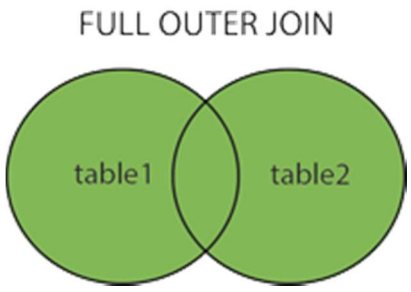
97.	**MIN MAX**	`SELECT MIN(Price) AS SmallestPrice`
		`FROM Products;`
		
		`SELECT MAX(Price) AS LargestPrice`
		`FROM Products;`

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

99.	**LEFT JOIN**	`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.
		
		 

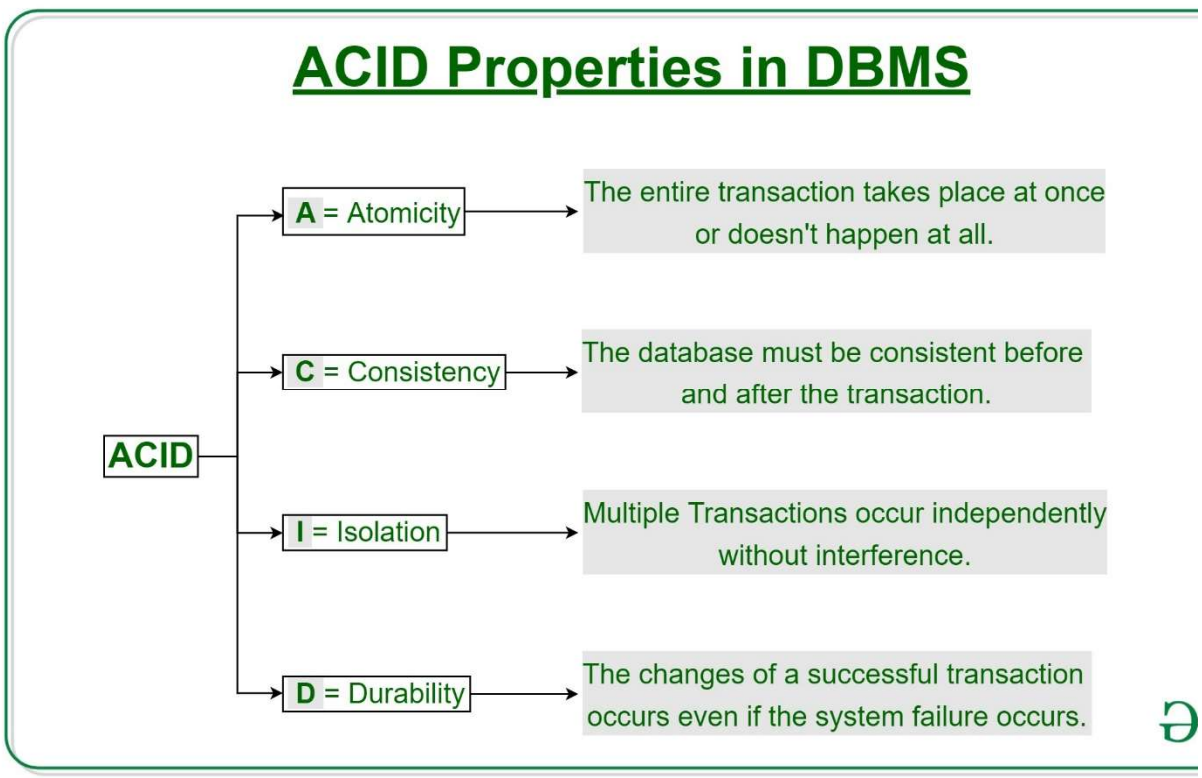
100.	**RIGHT JOIN**	`SELECT column_name(s)`
		`FROM table1`
		`RIGHT JOIN table2`
		`ON table1.column_name = table2.column_name;`
		
		Note: In some databases `RIGHT JOIN` is called `RIGHT OUTER JOIN`.
		
		 

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

		
102.	**SELF JOIN**	A self join is a regular join, but the table is joined with itself.
		The `TRUNCATE TABLE` command deletes the data inside a table, but not the table itself. `TRUNCATE TABLE Categories;`

103.	**GRANT**	`grant privilege_name on object_name`
		`to {user_name | public | role_name}`
		
		`grant insert,`
		`select on accounts to Ram`
		
		SQL Grant command is specifically used to provide privileges to database objects for a user. This command also allows users to grant permissions to other users too.

104.	**REVOKE**	`revoke privilege_name on object_name`
		`from {user_name | public | role_name}`
		
		`revoke insert,`
		`select on accounts from Ram`
		
		Revoke command withdraw user privileges on database objects if any granted

105.	**ACID properties**	
106.	**Types of pointers**	1. Null pointer. 2. Void pointer. 3. Wild pointer.

<table>
<tr><td></td><td></td><td>4. Dangling pointer.
5. Complex pointer.
6. Near pointer.
7. Far pointer.
8. Huge pointer.</td></tr>
<tr><td>107.</td><td>Structure & union</td><td>Differences:

<table><tr><th></th><th>STRUCTURE</th><th>UNION</th></tr><tr><td>Keyword</td><td>The keyword struct is used to define a structure</td><td>The keyword union is used to define a union.</td></tr><tr><td>Size</td><td>When a variable is associated with a structure, the compiler allocates the memory for each member. The size of structure is greater than or equal to the sum of sizes of its members.</td><td>when a variable is associated with a union, the compiler allocates the memory by considering the size of the largest memory. So, size of union is equal to the size of largest member.</td></tr><tr><td>Memory</td><td>Each member within a structure is assigned unique storage area of location.</td><td>Memory allocated is shared by individual members of union.</td></tr><tr><td>Value Altering</td><td>Altering the value of a member will not affect other members of the structure.</td><td>Altering the value of any of the member will alter other member values.</td></tr><tr><td>Accessing members</td><td>Individual member can be accessed at a time.</td><td>Only one member can be accessed at a time.</td></tr><tr><td>Initialization of Members</td><td>Several members of a structure can initialize at once.</td><td>Only the first member of a union can be initialized.</td></tr></table></td></tr>
<tr><td>108.</td><td>Malloc()</td><td>The “malloc” or “memory allocation” method in C is used to dynamically allocate a single large block of memory with the specified size. It returns a pointer of type void which can be cast into a pointer of any form. It doesn’t Iniatialize memory at execution time so that it has initializes each block with the default garbage value initially.
Syntax:

<code>ptr = (cast-type*) malloc(byte-size)</code></td></tr>
<tr><td>109.</td><td>Calloc()</td><td>1. “calloc” or “contiguous allocation” method in C is used to dynamically allocate the specified number of blocks of memory of the specified type. it is very much similar to malloc() but has two different points and these are:
2. It initializes each block with a default value ‘0’.
3. It has two parameters or arguments as compare to malloc().
Syntax:

<code>ptr = (cast-type*)calloc(n, element-size);</code></td></tr>
<tr><td>110.</td><td>Free()</td><td>“free” method in C is used to dynamically de-allocate the memory. The memory allocated using functions malloc() and calloc() is not de-allocated on their own. Hence the free() method is used, whenever the dynamic memory allocation takes place. It helps to reduce wastage of memory by freeing it.
Syntax:

<code>free(ptr);</code></td></tr>
<tr><td>111.</td><td>Realloc()</td><td>“realloc” or “re-allocation” method in C is used to dynamically change the memory allocation of a previously allocated memory. In other words, if the memory previously allocated with the help of malloc or calloc is insufficient, realloc can be used to dynamically re-allocate memory. re-allocation of memory maintains the already present value and new blocks</td></tr>
</table>

		will be initialized with the default garbage value. **Syntax:** `ptr = realloc(ptr, newSize);`

112.	**Call by value**	`#include<stdio.h>`
		`void change(int num) {`
		`  printf("Before adding value inside function num=%d \n",num);`
		`  num=num+100;`
		`  printf("After adding value inside function num=%d \n", num);`
		`}`
		`int main() {`
		`  int x=100;`
		`  printf("Before function call x=%d \n", x);`
		`  change(x);//passing value in function`
		`  printf("After function call x=%d \n", x);`
		`return 0;`
		`}`
		Output
		`Before function call x=100`
		`Before adding value inside function num=100`
		`After adding value inside function num=200`
		`After function call x=100`

113.	**Call by reference**	`#include<stdio.h>`
		`void change(int *num) {`
		`  printf("Before adding value inside function num=%d \n",*num);`
		`  (*num) += 100;`
		`  printf("After adding value inside function num=%d \n", *num);`
		`}`
		`int main() {`
		`  int x=100;`
		`  printf("Before function call x=%d \n", x);`
		`  change(&x);//passing reference in function`
		`  printf("After function call x=%d \n", x);`
		`return 0;`
		`}`
		Output
		`Before function call x=100`
		`Before adding value inside function num=100`
		`After adding value inside function num=200`
		`After function call x=200`

