**What is javascript?**

**JavaScript** is a high-level, dynamic, untyped, and interpreted programming language. It has been standardized in the ECMAScript language specification. Alongside HTML and CSS, it is one of the three essential technologies of World Wide Web content production; the majority of websites employ it and it is supported by all modern Web browsers without plug-ins.

**What is JQuery?**

**jQuery** is a fast, small, and feature-rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers. With a combination of versatility and extensibility, jQuery has changed the way that millions of people write JavaScript.

**Other Libraries:**

* JavaScriptMVC
* Meteor
* MooTools

**What is CSS?**

**Cascading Style Sheets** (**CSS**) is a style sheet language used for describing the presentation of a document written in a markup language.

**HTML vs HTML5**

HTML 1.0 - 4.0 (and XHTML) were all specifications for laying out text and images on a page. HTML5 includes not only laying out text and images, but also playing video and audio, interactive 2d and 3d graphics.New tags are audio, video, canvas etc.

**HTTP**

Short for **Hyper Text Transfer Protocol, HTTP** is the underlying protocol used by the World Wide Web. HTTP defines how messages are formatted and transmitted, and what actions Web servers and browsers should take in response to various commands

**Request vs Response**

**Request :** From Client to Server  
**Response:** From Server to Client  
**Server:** Receive Request and Send Response  
**Client:** Send Request and Receive Response

**Get vs Post**

**GET** - Requests data from a specified resource

**POST** - Submits data to be processed to a specified resource

**Some other notes on GET requests:**

* GET requests can be cached
* GET requests remain in the browser history
* GET requests can be bookmarked
* GET requests should never be used when dealing with sensitive data
* GET requests have length restrictions
* GET requests should be used only to retrieve data

**Some other notes on POST requests:**

* POST requests are never cached
* POST requests do not remain in the browser history
* POST requests cannot be bookmarked
* POST requests have no restrictions on data length

**Thin Client vs Thick Client**

A **thin client** is a stateless desktop terminal that has no hard drive. All features typically found on the desktop PC, including applications, sensitive data, memory, etc., are stored back in the data center when using a **thin client**.

A **fat client** (also called heavy, rich or **thick client**) is a computer (**client**) that typically provides rich functionality independent of the central server.

**Web Server**

A **web server** is an information technology that processes requests via HTTP

**IIS**

**Internet Information Services** (IIS, formerly **Internet Information Server**) is an extensible web server created by Microsoft for use with Windows NT family.[2] IIS supports HTTP, HTTPS, FTP, FTPS, SMTP and NNTP.

**Apache**

The Apache HTTP Server, is the world's most used web server software

**Apache Tomcat**

Apache Tomcat™ is an open source software implementation of the Java Servlet, JavaServer Pages, Java Expression Language and Java WebSocket technologies.

**Server-side Programming**

Server-side programming, is the general name for the kinds of programs which are run on the **Server**.

### Example Languages

* PHP
* Python
* ASP.Net in C#, C++, or Visual Basic.

**Client-side programming**

Much like the server-side, Client-side programming is the name for all of the programs which are run on the **Client**.

### Example languages

* JavaScript (primarily)

**what are webservices?**

The term Web services describes a standardized way of integrating Web-based applications using the XML, SOAP, WSDL and UDDI open standards over an Internet protocol backbone.

**Web 2.0**

The second stage of development of the Internet, characterized especially by the change from static web pages to dynamic or user-generated content and the growth of social media.

**Ajax**

method of exchanging data with a server, and updating parts of a web page - without reloading the entire page.

**what is REST services?**

REST (REpresentational State Transfer) is an architectural style, and an approach to communications that is often used in the development of Web services.

**SQL**

**Exists Statement**

The SQL EXISTS condition is used in combination with a subquery and is considered to be met, if the subquery returns at least one row. It can be used in a SELECT, INSERT, UPDATE, or DELETE statement.

When a subquery is introduced with the keyword EXISTS, the subquery functions as an existence test. The WHERE clause of the outer query tests whether the rows that are returned by the subquery exist. The subquery does not actually produce any data; it returns a value of TRUE or FALSE.

*SELECT \**

*FROM suppliers*

*WHERE EXISTS (select \* from orders*

*WHERE suppliers.supplier\_id = orders.supplier\_id);*

**IN Statement**

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

**Distinct Keyword**

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

**Like Operator**

The SQL **LIKE** clause is used to compare a value to similar values using wildcard operators. There are two wildcards used in conjunction with the **LIKE** operator:

* The percent sign (%)
* The underscore (\_)

**Sub Query**

A **Subquery** or Inner query or Nested query is a query within another SQL query and embedded within the WHERE clause.

A **subquery** is used to return data that will be used in the main query as a condition to further restrict the data to be retrieved.

**Subqueries** can be used with the SELECT, INSERT, UPDATE, and DELETE statements along with the operators like =, <, >, >=, <=, IN, BETWEEN etc.

**Group Functions (Sum, Avg, Max, Min, Count)**

**Group functions** produce results based on a group or set of rows

**Count(\*) vs Count(column)**

* Count(\*) counts all rows including null
* Count(coulumn) counts only non-null rows

**Having vs Where**

WHERE clause introduces a condition on individual rows; HAVING clause introduces a condition on aggregations

**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.

**Self Join**

A **self-join** is a query in which a table is joined (compared) to itself.

**LEFT JOIN**

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.

**RIGHT JOIN**

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.

**Full JOIN**

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.

**CROSS JOIN**

The **CARTESIAN JOIN** or **CROSS JOIN** returns the Cartesian product of the sets of records from the two or more joined tables. Thus, it equates to an inner join where the join-condition always evaluates to True or where the join-condition is absent from the statement.

**Syntax:**

The basic syntax of **CARTESIAN JOIN** or **CROSS JOIN** is as follows:

*SELECT table1.column1, table2.column2...*

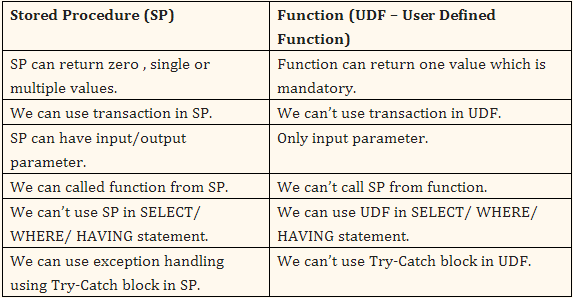
*FROM table1, table2 [, table3 ]*

**IF Statement**

In SQL Server, the **IF...ELSE** statement is used to execute code when a condition is TRUE, or execute different code if the condition evaluates to FALSE.

**Stored Procedure vs Function**

**Stored Procedures** are pre-compile objects which are compiled for first time and its compiled format is saved which executes (compiled code) whenever it is called. But **Function** is compiled and executed every time when it is called.



**Views**

A **view** is a virtual table.

**SQL CREATE VIEW Syntax**

*CREATE VIEW view\_name AS  
SELECT column\_name(s)  
FROM table\_name  
WHERE condition*

**SQL CREATE OR REPLACE VIEW Syntax**

*CREATE OR REPLACE VIEW view\_name AS  
SELECT column\_name(s)  
FROM table\_name  
WHERE condition*

**SQL DROP VIEW Syntax**

*DROP VIEW view\_name*

**Data Definition Language**

The **CREATE TABLE** statement is used to create a table in a database.Tables are organized into rows and columns; and each table must have a name.

**SQL CREATE TABLE Syntax**

*CREATE TABLE*table\_name *(*column\_name1 data\_type*(*size*),*column\_name2 data\_type*(*size*),*column\_name3 data\_type*(*size*),  
....  
);*

**SQL CREATE DATABASE Syntax**

*CREATE DATABASE*dbname*;*

**ALTER TABLE**

The **ALTER TABLE** statement is used to add, delete, or modify columns in an existing table.

**SQL ALTER TABLE Syntax**

To add a column in a table, use the following syntax:

*ALTER TABLE table\_name  
ADD column\_name datatype*

To delete a column in a table, use the following syntax (notice that some database systems don't allow deleting a column):

*ALTER TABLE table\_name  
DROP COLUMN column\_name*

To change the data type of a column in a table, use the following syntax:

**SQL Server / MS Access:**

*ALTER TABLE table\_name  
ALTER COLUMN column\_name datatype*

**My SQL / Oracle (prior version 10G):**

*ALTER TABLE table\_name  
MODIFY COLUMN column\_name datatype*

**Oracle 10G and later:**

*ALTER TABLE table\_name  
MODIFY column\_name datatype*

**DROP TABLE**

The DROP TABLE statement is used to delete a table.

*DROP TABLE table\_name*

**TRUNCATE TABLE**

If we only want to delete the data inside the table, and not the table itself Then, use the **TRUNCATE TABLE** statement:

*TRUNCATE TABLE table\_name*

**SQL SELECT**

The SELECT statement is used to select data from a database.The result is stored in a result table, called the result-set.

**SQL SELECT Syntax**

*SELECT*column\_name*,*column\_name *FROM*table\_name*;*

**SQL UPDATE**

The **UPDATE** statement is used to update existing records in a table.

**SQL UPDATE Syntax**

*UPDATE table\_name  
SET*column1*=*value1*,*column2*=*value2*,...  
WHERE*some\_column*=*some\_value*;*

**SQL DELETE Statement**

The DELETE statement is used to delete rows in a table.

**SQL DELETE Syntax**

*DELETE FROM*table\_name *WHERE*some\_column*=*some\_value*;*

**SQL INSERT INTO**

The INSERT INTO statement is used to insert new records in a table.

**SQL INSERT INTO Syntax**

*INSERT INTO*table\_name*(*column1*,*column2*,*column3*,...)  
VALUES (*value1*,*value2*,*value3*,...);*

**SQL UPDATE**

The UPDATE statement is used to update existing records in a table.

**SQL UPDATE Syntax**

*UPDATE*table\_name *SET*column1*=*value1*,*column2*=*value2*,...  
WHERE*some\_column*=*some\_value*;*

**Primary Key**

* Primary Key can't accept null values.
* We can have only one Primary key in a table.
* Primary key can be made foreign key into another table.

**Unique Key**

* Unique key can accept only one null value.
* We can have more than one unique key in a table.
* Primary key can be made foreign key into another table.

**Foreign Key**

A **foreign key** is a column (or columns) that reference a column (most often the primary **key**) of another table. The purpose of the **foreign key** is to ensure referential integrity of the data. In other words, only values that are supposed to appear in the database are permitted.

**CHAR vs VARCHAR vs NVARCHAR**

**CHAR** Data Type is a Fixed Length Data Type. For example if you declare a variable/column of CHAR (10) data type, then it will always take 10 bytes irrespective of whether you are storing 1 character or 10 character in this variable or column. And in this example as we have declared this variable/column as CHAR(10), so we can store max 10 characters in this column.

On the other hand **VARCHAR** is a variable length Data Type. For example if you declare a variable/column of VARCHAR (10) data type, it will take the no. of bytes equal to the number of characters stored in this column. So, in this variable/column if you are storing only one character then it will take only one byte and if we are storing 10 characters then it will take 10 bytes.

**NCHAR vs NVARCHAR**

It can store Unicode charachters, so you can in one column have multiple languages and letters (latin, cyrilic, arab etc). Because it is unicode each charachter consumes 2 bytes of memory. **Nchar** is fixed length and **nvarchar** is variable length - same as above.

**IDENTITY Column**

An **identity column** has a name, initial seed and step.  When a row is inserted into a table the column will take the value of the curent seed incremented by the step.

**Cluster vs non cluster Index**

With a clustered index the rows are stored physically on the disk in the same order as the index. There can therefore be only one clustered index.

With a non clustered index there is a second list that has pointers to the physical rows. You can have many non clustered indexes, although each new index will increase the time it takes to write new records.

**Normalization**

**1NF**

* Eliminate the Repeating Groups
* Identify the Primary Key
* Identify all Dependencies

**2NF**

Eliminate all partial dependencies from the 1NF format

* Identify All the Key Components
* Identify the Dependent Attributes

**3NF**

Eliminate all transitive dependencies from the 2NF format.

* Identify Each New Determinant
* Identify the Dependent Attributes