This is Salesforce Object Query Language designed to work with SFDC Database. It can search a record on a given criteria only in single sObject.

Like SOSL, it cannot search across the multiple objects but it does support nested queries.

## SOQL Example

Consider our ongoing example of Chemical Company. Suppose, we would like to have a list of records which are created today and whose customer name is not 'test'. In this case, we will have to use the SOQL query as given below:

**Example:**

```
//fetching the Records via SOQL
List<apex_invoice__c> InvoiceList = new List<apex_invoice__c>();
InvoiceList = [SELECT Id, Name, APEX_Customer__r.Name, APEX_Status__c FROM
APEX_Invoice__c WHERE createdDate = today AND APEX_Customer__r.Name != 'Test'];//SOQL
query for given  criteria

//Printing the fetched records
System.debug('We have total '+InvoiceList.size()+' Records in List');
for (APEX_Invoice__c objInvoice: InvoiceList) {
 System.debug('Record Value is '+objInvoice);//Printing the Record fetched
}
```

You could run the SOQL query via Query Editor in Developer console as shown below.

Run the below query in developer console. Search for the Invoice records created today.

```
SELECT Id, Name, APEX_Customer__r.Name, APEX_Status__c FROM APEX_Invoice__c WHERE
createdDate = today
```

You must select the fields for which you would like to have values for, otherwise, it can throw run time errors.

## Traversing Relationship Fields

This is one of the most important part in SFDC as many times we need to traverse through the parent child object relationship.

Also, there may be cases when you would need to insert two associated objects records in Database. For example, Invoice object has relationship with Customer object and hence one Customer can have many invoices.

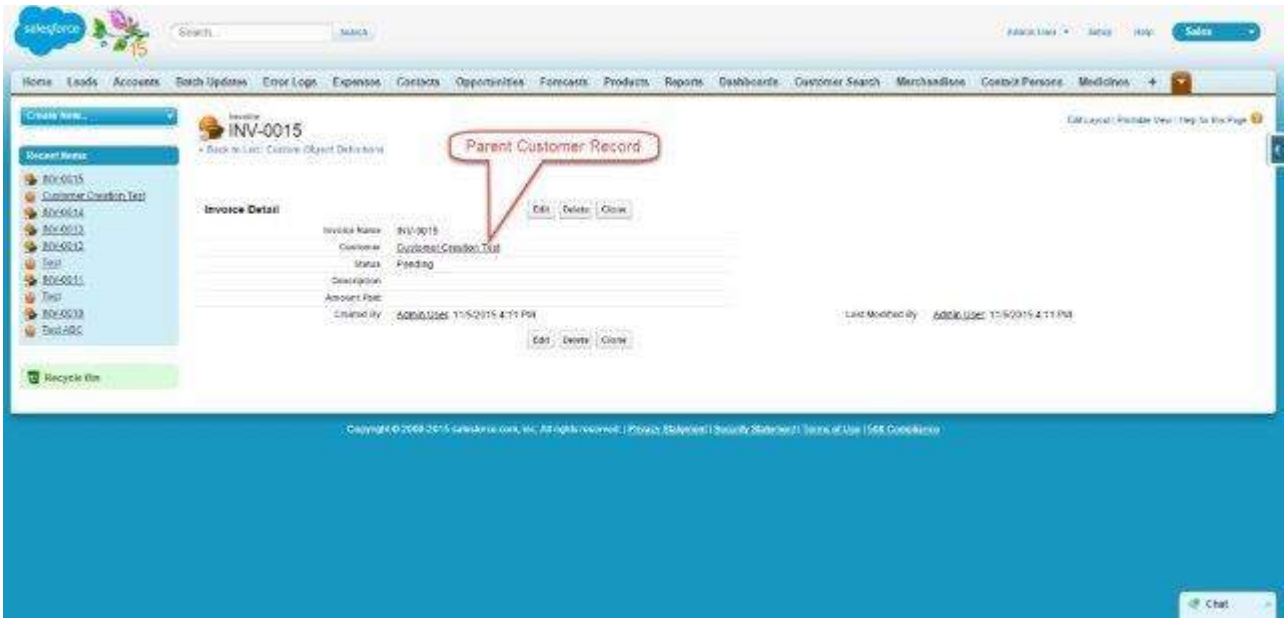
Suppose, you are creating the invoice and then you would like to relate this invoice to Customer. You can use the following code for this functionality:

```
//Now create the invoice record and relate it with the Customer object
//Before executing this, please create a Customer Records with Name 'Customer Creation
Test'
APEX_Invoice__c objInvoice = new APEX_Invoice__c();

//Relating Invoice to customer via id field of Customer object
objInvoice.APEX_Customer__c = [SELECT id FROM APEX_Customer__c WHERE Name = 'Customer
Creation Test' LIMIT 1].id;
objInvoice.APEX_Status__c = 'Pending';
insert objInvoice;//Creating Invoice
System.debug('Newly Created Invoice'+objInvoice);//Newly creted invoice
```

Execute this code snippet in Developer Console. Once executed, copy the Id of invoice from

Developer console and then open the same in SFDC as shown below. You can see that the Parent record has already been assigned to Invoice record as shown below.




## Fetching Child Records

Let's take an example where we would like to have all the invoices related to particular customer record. For this, you must know the child relationship name. To see the child relationship name, go to the field detail page on child object and check the "Child Relationship" value. In our example, it is invoices appended by __r at the end.

**Example:**

First of all we will need to setup data, create a customer with name as 'ABC Customer' record and then add 3 invoices to that customer.

Now, we would like to fetch that how many invoices the Customer 'ABC Customer' has. Below is the query for the same:

```
//Fetching Child Records using SOQL
List<apex_customer__c> ListCustomers = [SELECT Name, Id, (SELECT id, Name FROM
Invoices__r) FROM APEX_Customer__c WHERE Name = 'ABC Customer'];//Query for fetching the
Child records along with Parent
System.debug('ListCustomers '+ListCustomers);//Parent Record

List<apex_invoice__c> ListOfInvoices = ListCustomers[0].Invoices__r;//By this notation,
you could fetch the child records and save it in List
System.debug('ListOfInvoices values of Childs '+ListOfInvoices);//Child records
```

You could see the Record values in Debug logs.

## Fetching Parent Record

Suppose, you would like to fetch the Customer Name of Invoice whose created date is today, then you could use the below query for the same:

**Example:** Fetch the Parent record's value along with the child object.

```
//Fetching Parent Record Field value using SOQL
List<apex_invoice__c> ListOfInvoicesWithCustomerName = new List<apex_invoice__c>();
ListOfInvoicesWithCustomerName = [SELECT Name, id, APEX_Customer__r.Name FROM
APEX_Invoice__c LIMIT 10];    //Fetching the Parent record's values
for (APEX_Invoice__c objInv: ListOfInvoicesWithCustomerName) {
 System.debug('Invoice Customer Name is '+objInv.APEX_Customer__r.Name); //Will print
the values, all the Customer Records will be printed
}
```

Here we have used the notation APEX_Customer__r.Name, where APEX_Customer__r is parent relationship name, here you have to append the __r at the end of the Parent field and then you could fetch the parent field value.

## Aggregate Functions

SOQL does have aggregate function as we have in SQL. Aggregate functions allow us to roll up and summarize the data. Let's have a details look on the same.

Suppose, you wanted to know that what is the average revenue we are getting from Customer 'ABC Customer', then you could use this function to take up the average.

**Example:**

```
//Getting Average of all the invoices for a Perticular Customer
AggregateResult[] groupedResults = [SELECT AVG(APEX_Amount_Paid__c)averageAmount FROM
APEX_Invoice__c WHERE APEX_Customer__r.Name = 'ABC Customer'];
Object avgPaidAmount = groupedResults[0].get('averageAmount');
System.debug('Total Average Amount Received From Customer ABC is '+avgPaidAmount);
```

Check output in Debug logs. Note that any query that includes an aggregate function returns its results in an array of AggregateResult objects. AggregateResult is a read-only sObject and is only used for query results. It is useful when we would like to generate the Report on Large data.

There are other aggregate functions as well which you could use to perform data summary.

**MIN**-This can be used to find the minimum value

**MAX**-This can be used to find the maximum value.

## Binding Apex Variables

You could use the Apex variable in SOQL query to fetch the desired results. Apex variables can be referenced by the Colon : notation.

**Example:**

```
//Apex Variable Reference
String CustomerName = 'ABC Customer';
List<apex_customer__c> ListCustomer = [SELECT Id, Name FROM APEX_Customer__c WHERE Name
= :CustomerName];//Query Using Apex variable
System.debug('ListCustomer Name'+ListCustomer);//Customer Name
```