We have now FIELDS(ALL), FIELDS(STANDARD), and FIELDS(CUSTOM) which we can use in the SOQL query.

**FIELDS(ALL)** – This fetches all the fields of an object. This is similar like Select \* from SQL.

**FIELDS(STANDARD)** – This fetches all standard fields of an object.

**FIELDS(CUSTOM)** – This fetches all custom fields of an object.

## **Select FIELDS(ALL) from Account LIMIT 100**

FIELDS(ALL) is just like ‘\*’ in SQL. It fetches all the columns(fields).

## **Select FIELDS(Standard) from Contact LIMIT 50;**

FIELDS(Standard) will fetch all the standard fields.

## **What is bounded and unbounded query?**

When **API cannot determine the set of fields in advance**, then it will consider query as **Unbounded Query.**

If **API can determine the set of fields in advance**, then it is **Bounded Query**.

For example – The number of custom fields for an object is not pre-determined, so **FIELDS(CUSTOM) and FIELDS(ALL) are considered as Unbounded Query.**

## **How to prevent SOQL injections?**

There are a number of techniques you can use to prevent SOQL injection:

* **Static queries with bind variables**
* **String.escapeSingleQuotes()**
* **Type casting**
* **Replacing characters**
* **Whitelisting**

## **What is disadvantage of Dynamic SOQL?**

One disadvantage with DYNAMIC SOQL is it causes SOQL injection in where condition which fetching on the basis of some text. To avoid which we need to use String.escapeSingleQuotes. There is no possibility of these in Static SOQL.

## **When do we use database.query()?**

database.query allows you to make a dynamic SOQL query at runtime. You can build up a string and then use that as a query string at run time in the database.

Query statement to make a SOQL call that is determined at run time.

The database.query method should be used in instances where you are wanting to do a dynamic runtime SOQL query for your code.

## **When do we use database.getquerylocator()?**

Database.getQueryLocator returns a Query Locator that runs your selected SOQL query returning list that can be iterated over in batch apex or used for displaying large sets in VF (allowing things such as pagination). The query locator can return up to 50 million records and should be used in instances where you want to batch a high volume of data.

## **ORDER BY CLAUSE**

**[ORDER BY fieldOrderByList {ASC|DESC} [NULLS {FIRST|LAST}] ]**

**NULLS FIRST** or **NULLS LAST** orders by Null values first or at last. By default Null values comes first.

### **SELECT Name FROM Account ORDER BY Name DESC NULLS LAST**

## **What is OFFSET in soql?**

A. Offset is used to skip the rows from the start of the query.

OFFSET considerations:

* the maximum limit of offset is 2000 rows. requesting on offset greater than 2000 will result in a number-out-side-valid-range error.
* offset can’t be used in subquery in the where clause.

**Syntax to fetch the records from recycle bin using soql?**  
A. **SELECT Id, isDeleted FROM <Oblectname> WHERE isDeleted = true All ROWS** – This will only return the deleted rows.  
**SELECT Id, isDeleted FROM <Oblectname> WHERE isArchived = true All ROWS** – This will only return the archived rows.  
**SELECT Id, isDeleted FROM <Oblectname> All ROWS** – This will return the deleted records, archived records and records that are neither deleted nor archived (data set identical to the one returned by a SOQL not using ALL ROWS) as well.

**You can-not use ALL ROWS and FOR UPDATE together.**

## **What is FOR UPDATE clause in soql?**

A. For Update clause will lock the records from getting updated from other transactions untill the current transaction is completed.

syntax: [select id, name from account for update]

## **How you can use Datetime field as a criteria in SOQL Query ?**

We cannot use Datetime as condition in Where Clause in between single Quotes.

You can do something like this ,

WHERE CreatedDate > 2005-10-08T00:00:00Z

Or, you can also use Date Literals like

WHERE CreatedDate > YESTERDAY

**Or, if working in apex, we can use a bind variable like:**

**Date d = System.today().addMonths(-3);**

**SELECT Id, Name from Account WHERE createdDate > :d**

**The above query can used to fetch the accounts which created within past three months.**

## **Select id from Account ALL ROWS. What is result of the query?**

SOQL statements can use the ALL ROWS keywords to query all records in an organization, including deleted records and archived activities.

System.assertEquals(3, [SELECT COUNT() FROM Contact WHERE AccountId = a.Id ALL ROWS]);

You can use ALL ROWS to query records in your organization’s Recycle Bin. You cannot use the ALL ROWS keywords with the FOR UPDATE keywords.

## **SOQL For Loops Versus Standard SOQL Queries?**

SOQL query sometimes returns so many sObjects that the limit on heap size is exceeded and an error occurs. To resolve, use a SOQL query for loop instead, since it can process multiple batches of records by using internal calls to query and queryMore.

Developers should always use a SOQL for loop to process query results that return many records, to avoid the limit on heap size.

## **What is polymorphic relationship?**

A polymorphic relationship is a relationship between objects where a referenced object can be one of several different types. For example, the Who relationship field of a Task can be a Contact or a Lead.

**Approach 1**: Filter results using the Type qualifier.

Example: Events that are related to an Account or Opportunity via the What field.

List<Event> events = [SELECT Description FROM Event WHERE What.Type IN (‘Account’, ‘Opportunity’)]

**Approach 2**: Use the TYPEOF clause in the SOQL SELECT statement.

Example : Query Events that are related to an Account or Opportunity via the What field.

List<Event> events = [SELECT TYPEOF What WHEN Account THEN Phone WHEN Opportunity THEN Amount END FROM Event];

These queries return a list of sObjects where the relationship field references the desired object types.