Q1 Why do we use DML statements?

Ans: DML **provides a straightforward way to manage records by providing simple statements to insert, update, merge, delete, and restore records**. Because Apex is a data-focused language and is saved on the Lightning Platform, it has direct access to your data in Salesforce.

Q2 What is the use of upsert statements? And how does it work?

Ans: Using the upsert operation, you can either **insert or update an existing record in one call**. To determine whether a record already exists, the upsert statement or Database method uses the record's ID as the key to match records, a custom external ID field, or a standard field with the idLookup attribute set to true.

Q3 What is the usage of a merge statement?

Ans: You can pass a master record and up to two additional sObject records to a single merge method. Using the Apex merge operation, **field values on the master record always supersede the corresponding field values on the records to be merged**.

Q4 Merge statement is applicable on which objects?

Ans: Contacts, lead and Account**.**

Q5 Why do we use the Undelete statement?

Ans: While the records are still in the Recycle Bin, you can restore them using the undelete operation. This is useful, for example, **if you accidentally deleted some records that you want to keep**.

Q6. Give a difference between upsert and insert statement.

Ans: Upsert stands for both update and insert.

Insert is a dml statement used to insert the records in to the object.

Q7. Write a governor limit for using Dml statements.

Ans: 150

Q8. What is the purpose of External id when using Upsert statement.

Ans: Use of upsert with an external ID can **reduce the number of DML statements in your code, and help you to avoid hitting governor limits.**

External Id is **a unique key or primary key that is different from salesforce Id**. It is slightly created by the user in order to insert, upsert, delete and update records in Salesforce. External Id cannot be duplicated

Questions on Try

Q1. Can we have a try block without a catch block?

Ans: Yes, It is possible to have a try block without a catch block by using a final block.

Q2. Can a try block have multiple catch blocks?

Ans: One TRY statement can have multiple CATCH blocks for different exception types.

Q3. Which class is there to handle DML related exceptions?

Ans: Using try catch blocks

Q4. How many types of exceptions in class?

Ans: We have 3 type of exception class

1. **DML Exception-** insert, update delete
2. **General Exception-** Nullpointer exception, sObject, String to integer
3. **User define exception-** THROW()

Q5. What is the return type of database.delete() method?

Ans: **DeleteResult objects** is returned with the delete database method.

Q6. What is the return type of database.insert() method?

Ans: When you use the database method for an insert or update DML operation, **an array of SaveResult** is returned.

Q7. Explain Getmessage, getlinenumber, getfields and getstatuscode.

Ans: Getfields(): Returns an array of one or more field names. Identifies which fields in the object, if any, affected the error condition.

Getlinenumber(): Returns the line number from where the exception was thrown.

Getmessage(): Returns the error message text.

Getstatuscode(): Returns a code that characterizes the error.

Q8. What is the purpose of Finally block?

Ans: The finally statement **identifies a block of code that is guaranteed to execute and allows you to clean up your code**. A single try statement can have up to one associated finally statement. Code in the finally block always executes regardless of whether an exception was thrown or the type of exception that was thrown.