ETL - Extraction Transformation Loading.

**Good practices:**

1. Write comments for the queries.
2. Normalize your tables to the best level.
3. Make use of the default permissions.
4. Make sure that the structure of the database is optimized.

**How to design a database?**

**Normalization:**

**1 Normal Form:**

* Each column should have just one value.
* Values in the column should be uniquely identified.
* Each row should be uniquely identified- Add unique id or more columns to make unique. It means that if the table do not have a unique id then we can add a unique id to it.

**2 Normal Form:**

* No partial dependency – Each column should be dependent on the key.
* To be in second normal form, a relation must be in first normal form and relation must not contain any partial dependency. A relation is in 2NF iff it has **No Partial Dependency,**i.e.**,**no non-prime attribute (attributes which are not part of any candidate key) is dependent on any proper subset of any candidate key of the table.

**3 Normal Form:**

* All the columns can only be determined by the key and no other column.

**4 Normal Form:**

* There should not be any multivalued dependency.

**Problem Statement:**

An affiliation body wants an online platform where users can apply to become training partners or assessment partners.

Affiliation body want the system to allow them to view the application submitted by the user before they can take a decision on approval or rejection on it, also they want a user to be able to edit application after submission if affiliation body asks so.

After approval of application of a training partner, they can add details of batches of candidates they have trained and admin can assign any approved assessment partner to assess the batch.

**Alter tables:**

Alter table <table\_name> ADD|DROP|MODIFY|RENAME <column definition>’

alter table user add (pan\_number int);

desc user;

create table profile(

dob date,

name varchar(10),

appliation\_id varchar(6),

id varchar(6)

);

alter table profile add column (pan\_number int,

state varchar(20),

district varchar(20));

insert into user values(“[lohit@tothenew.com](mailto:lohit@tothenew.com)”,”yatin@123”,2,1);

insert into user values(“[deepshika@tothenew.com](mailto:deepshika@tothenew.com)”,”deeps@123”,2,2);

insert into user values(“[rohit@tothenew.com](mailto:rohit@tothenew.com)”,”rohit@123”,1,3);

insert into user values(“souvik[@tothenew.com](mailto:lohit@tothenew.com)”,”souvik@123”,1,4);

**Select:**

Select <comma separated list of columns> from <table\_name> where <exp>;

First the select statement goes to the where clause and applies it to the table and then it fetches the columns which are selected by us.

**Like Clause:**

Create table <new\_table> like <original\_table>;

Create table application\_replica like application;

Create table batch\_copy like batch;

Create table role\_duplicate like role;

**Copy data from one table to another table:**

Insert into <name of table> select <list of colums> from <name of source table> where expression;

Insert into batch\_copy select \* from batch;

**Exercise:**

**Create a table like user; backup the users registered in year 2017.**

**Solution:**

Create table user\_copy like user;

Alter table user\_copy add date\_of\_application date;

Insert into user\_copy select \* from user where date\_of\_application >“2017/1/1” and date\_of\_application<”2017/12/31”;

**Wild Card Characters:**

**% -** Can be used in place of a set of characters in a string. Represents 0 or many characters.

**\_ -** Represents only one characters.

**When we use wild card characters then we have to use like instead of =.**

**Select all the users with username that ends with ‘lm’.**

Select \* from user where username like ‘%m’;

**Select all the users who have 3 characters long username.**

Select \* from user where username like ‘\_\_\_’;

**Assignment:**