SQL Day1

--Java is mostly used to develop business applications

Business Organizations:

Objectives of any BO, profit.

1. small scale business organizations (grossary shop, petrol pump)

2. large scale business organizations (HDFC bank, Indian railway, SBI,PEPSI, MASTERCARD)

Enterprises.

These BO provides their services to the client/customer. and to computerized those services what ever application we develop is known as business application.

Common general things in any business organizations:

1. store and maintain business data in a secure and easily retrival manner.

2. processing that data according to the business rule.

3. presenting the data in user-understandable format.

Data and information:
==========

Data : it is a collection of raw and isolated facts.

Information : when we process the data , then we gets meaningfull results, this is called infromation.

Datastore:
---

it is a store where we can store or keep the data.

1. normal books and papers.

2. flat files in computer system(notepad, excel sheet, word files)

disadv of flat files:
---

1. data maintainace.

2. data redundency.
3. data integrity
4. security
5. data retrival

to overcome these problems we need to store the data inside the DBMS s/w (RBDMS s/w)

Database: it is a organized collection of interrelated data or structured collection of data.

DBMS is a type of s/w there we can mange multiple databases.

RDBMS: in this model, the data is stored in 2 dimentaional tables.

--we have multiple RDBMS s/w are there:

Mysql (Oracle)
Orcale s/w
postgres
ms-access
sql-server
db2
etc

RDBMS is an extension of DBMS s/w

Note: every RDBMS is a DBMS but reverse is not true.

--in order to work with RDBMS we need to use SQL (structured query language),it is an interface by using which we can work with any kind of RDBMS s/w.

--sql is a case insensetive language.

--sql language is a collection of predefined commads

these commands are categorized into following categories:

1. DDL (Data defination language)
(create, alter, drop, truncate, rename)

2. DML (Data manipulation Language)
(insert, update, delete)

3. DRL (Data Retrival language)
(select)

4. TCL (Transaction control language)
(commit, rollback, savepoint)

5. DCL (Decision Control language)
(grant revoke)

>show databases;

>create database sb101db;

after creating the db inside the mysql rdbms s/w we need to move inside that db.

>use sb101db;

>show tables;

>create table student(roll int,name varchar(12),marks int);

or

```
create table student
(
roll int,
name varchar(12),
marks int
);
```

>desc student;

>select * from student;

1. DDL (Data defination language)
(create, alter, drop, truncate, rename)
==========

Datatypes in mysql:
---

1. numeric types

2. string types

3. date and time types

1. numeric types
---

tinyint : 1byte
smallint --2 byte
mediumint - 3byte

int ---4byte
bigint--8byte

floating point:
---

float(6,2) :- the column can store 6 digit with 2 decimal places

2. string type:
---

1. char : fixed length of string range bt 0 to 255 char

2. varchar: variable length of string bt 1 to 65500, here we must define the length e

ex:

char(4)

varchar(4)

value	char(4)	storage_required
'a'	------->	4 bytes
'ab'	------->	4 bytes
'abcdef'	------> error data is too long	

value	varchar(4)	storage_required
'a'	------->	1 bytes
'ab'	------->	2 bytes
'abcdef'	------> error data is too long	

Note: in the term of efficiency , if we r storing string with variable lenght the we should use varchar, and if the length is always fixed then we should use char, here char is slightly faster than varchar.

3. date and time:
---

a. date yyyy-mm-dd

b. datetime : yyyy-mm-dd hh:mm:ss

(1.create, 2.alter, 3.drop, truncate, rename)

2.alter: --it is used to change the strcuture of the existing table

--this command has 4 sub commands :

a. add
b. modify
c. drop
d. change

a. add: it is used to add the new cols in the existing table

ex:
> alter table student add address varchar(15);

b. modify : it is used to change the col datatype of col size.

ex:

> alter table student modify address varchar(20);

c. drop : to drop a single or multiple columns

ex:

>alter table student drop column address;

d. change:

--to rename a column

ex:

> alter table student change name sname varchar(12);

3. drop : to drop entire table .

>drop table student;

Note: it can not be rolledback.

4. truncate : this command is used to truncate all the rows/records permanently from the table
--this command also can not be rolledback.

>truncate table student;

5.rename: it is used to rename table.

ex:

>rename table student to student1;

DML (insert,update,delete)

1. insert:
---

inserting all columns values:

>insert into student values(10,'ram',780);

inserting partial columns values:

>insert into student(roll,sname) values(10,'amit');

2. update:

--it is used to update the date within a table

ex1: updating all the marks for all the students.

> update student set marks=marks+50;

ex2: updating marks for only one student.

>update student set marks=marks+50 where sname='ram';

>update student set marks=marks+50 where marks > 800;

>update student set sname='mukesh' where roll=30;

3. delete:

--it is used to delete the records/rows from the table.

>delete from student; // it will delete entire record from the table like truncate command.

>delete from student where sname = 'amit';

DRL(select)
==========

--this command is used to quering a table.

syntax:

select col1,col2,..

```
from tablename
where condition
groupby colname
having condition
orderby colname [asc/desc]

ex1:

>select * from student;   // all the column and all the rows

ex2: restricting the rows by using where clause.

> select * from student where roll =10;

>select sname from student;

>select sname from student where roll =10;

using order by: to sort the record.
--------------------

> select * from student order by marks desc;

operators:
========

1.  Aritmatic operators: (*,/, + , -, %)


Note: mostly aritmatic operator r used after select statement (90%)
 and all other type of remaining operators r used in where clause only

2.  relational operators (= > < <= >= [!= or <>])

3.  logical operator (AND OR NOT)

4.  Special operator(IN, LIKE,..)


ex: Arithmatic operator:

> select sname,marks, marks+100 from student;

>select sname,marks, marks+100 Gracemarks from student;

Note: this temparory name of a column we can not use inside where clause.

ex:

> select sname,marks from student where roll != 10;

using distinct:-- getting unique data
------------------
```

```
>select distinct marks from student;


special operators:
---------------------

IN .... NOT IN

BETWEEN  ----> NOT BETWEEN

IS NULL  ----> IS NOT NULL

LIKE ---->  NOT LIKE


> select * from student where marks IS NULL;


>select * from student where marks BETWEEN 500 AND 800;
or
>select * from student where marks>=500 AND marks <=800;


LIKE ---> NOT LIKE

--it is used to retrive the data based on charecter pattern

1.  % -- it represents string or group of charecters.

2.  _  -- it represents a single charecter.


ex:

select * from student where sname LIKE 'r%';    : name should start with r

ex: in name r should be any charecter

>select * from student where sname LIKE '%r%';
```

SQL Day2

Constraints:

--constraints are created on the columns.

--it prevents invalid data entry into our tables.

3. not null

4. unique

5. Primary key

6. Foreign key

7. check : mysql does not support this check constraint.

Note: some contraints we can apply at column level and some constraints we can apply at table level.

column level : where we define the column
not null
unique
Primary key

table level : after defining all the columns
composit key (multi-column primary key)
foreign key

4. not null:

--null value is not allowed.

5. unique:

--here duplicate values are not allowed.

--here we can insert null values, multiple time.

Note:--whenever we define a unique then automatically DB engine will create an index on those column.

--so searching based on unique column will become fast.

6. Primary key:

--here also DB engine create index of that column.

--value can not be duplicate
--null is also not allowed.

another diff with PK and unique:- in one table we can have multiple unique constraints but in one table we can have only one PK.

--if we want to apply PK on multiple column then it will become composit key.

****Note: with the help of PK column we can uniquely identify one record of a table.

```
create table student
(
roll int primary key,
name varchar(12) not null,unique
address varchar(12) unique,
marks int
);
```

teacher(tname, subject, age, phone, email)

```
create table teacher
(
tname varchar(12),
phone varchar(10),
email varchar(18),
age int,
subject varchar(12),
Primary key(tname,phone)
);
```

here tname and phone will become a composit key, this combination can not be duplicate.

Foreign key:
==========

--with the help of FK we inforce the refrential integrity.
--with the help of FK we can establish relationship bt 2 tables.

--second table FK must refer to first table PK.

--PK related FK column must belongs to the same datatype but colmun name can be diff.

--FK can accept the duplicate and null value also.

```
create table dept
(
did int primary key,
dname varchar(12),
location varchar(12)
);

create table emp
(
eid int primary key,
ename varchar(12),
address varchar(12),
salary int,
deptid int, foreign key (deptid) references dept(did)
);


--with the help of FK we establish parent and child relationship among
tables.

here dept table will act as parent table
and emp table will act as a child table

--the table which contains the FK column will be considered as child
table.




Note: whenever we try to establish a relationship using FK then DB
violates following
7. rules:

3.  deletion or updation in parent table (even we can not drop the parent
    tables also)
4.  insertion in child table.




--to overcome this updation and deletion problem we should use

on delete cascade

or

on delete set null

simillarly for update also.

--while creating the child table.

create table emp
(
```

```
eid int primary key,
ename varchar(12),
address varchar(12),
salary int,
deptid int, foreign key (deptid) references dept(did) on delete cascade
);
```

adding a constraint to an existing table:

```
>create table a1(id int,name varchar(12));

> alter table a1 modify id int primary key;
```

adding foreign key:

```
>create table b1(bid int);

>alter table b1 add foreign key(bid) references a1(id) on delete set null;
```

functions in mysql:

--it is used to solve a perticular task.

--a sql function must return a value.

--in sql we have 2 types of functions:

6. predefined functions
7. user-defined functions(it is in PL/SQL)

predefined functions:

--it is devided into 4 categories:

6. number functions
7. charecter functions
8. group functions or aggregrate functions
9. date functions.

2. number functions:
---

4. abs(): it returns the absolute number.

ex:

```
>select abs(-40) from dual;   //40

5.  mod(m,n) : - it returns the reminder of m/n;

ex:

>select mod(10,2) from dual;

round(m,n)
truncate(m,n)

ex:

select round(12.43482,3) from dual;  //  12.435

select truncate(12.43482,3) from dual;  //  12.434


ceil()
floor()

greatest() least():
----------------------

--it will return biggest and smallest value from the list of arguments.

ex:

 select greatest(10,12,8,15) from dual; // 15


Note: from a single column if we want to max and min value then we should
use group functions
like max() min();


2.charecter functions:
---------------------------

3.  upper()
4.  lower()
5.  length()
6.  replace()
7.  concat()
8.  substr()


ex:-

>  select upper(ename) from emp where eid =100;

>select substr('ratan',3,2) from dual; // ta
```

9. date function:
---

4. sysdate() : it will return the current date and time;

ex:

>select sysdate() from dual;

5. date_format()

> select date_format(sysdate(), '%d %m %Y');

6. adddate()

syn:

adddate(date, INTERVAL value unit);

DAY
HOUR
YEAR
MONTH
WEEK

group function or aggregrate function:
==========

--these functions operates over several values of a single column and then result in a
single value.

c. max()
d. min()
e. avg()
f. sum()
g. count(*)
h. count(columnName)

group by clause:
==========

--the main purpose of group by clause is to group the records.

--this clause is mostly used with the group functions only.

--it is used to devide the similar data items into set of logical groups.

short syn:

```
select col_name from table group by col_name;

full syn:
----------

select col_names
from
tablename
[where condition]  ----opt
group by col_names
[having <cond>]----opt



 eid  | ename   | address | salary | deptid |
+------+---------+---------+--------+--------+
|  100 | ram     | pune    |   7800 |     10 |
|  102 | dinesh  | pune    |   7800 |     13 |
|  103 | manoj   | delhi   |   8500 |     13 |
|  104 | chandan | mumabi  |   8200 |     14 |
|  105 | manoj   | NULL    |   4500 |     14 |
| 1001 | ramesh  | patna   |   8800 |     14 |



--the above data is called as detailed data and after performing the group
by ,we get the
summerized data which is usefull for analysis.

mysql> select sum(salary) from emp; // it will calculate the salary from
whole table

>select deptid, sum(salary) from emp group by deptid; // dept wise total
salary


>select deptid,max(salary), min(salary), avg(salary) from emp group by
deptid;


rule:

e.  group functions we can not use inside the where clause.
f.  other than group function all the columns mentioned inside the select
    clause must be there after the group by clause otherwise (oracle db
    will give an error and mysql may give the unexpected result).

> select deptid,ename,max(salary) maximum, min(salary) minimum,
avg(salary) from emp group by deptid,ename;

Having:
----------
```

after group by clause we r not allowed to use where clause in place of where clause we should we having clause after group by clause.

with where clause:-
> select deptid,sum(salary) from emp where deptid IN(13,14) group by deptid having sum(salary) > 10000;

without using where clause:
>select deptid,sum(salary) from emp group by deptid having sum(salary) > 10000;

SQL Day3

Joins
=====

--Join is used to receive data from multiple tables or by using we can combine records from multiple tables.

there r following types of joins:

8. Inner Join
9. Outer Join
    - Left Outer join
    - Right Outer Join
    - Full Outer Join

10. self join

11. Cross Join (cartesion product)

Note: when we try to get the data from more than one table without using joining condition, then it is called cross join, in this case every record of the first table will be mapped with every record of the second table.

--with the cross join we don't get the meaningfull data, in order to get the meaningfull data we need to use other types of joins.

INNER JOIN:
==========

--here we need to apply joining condition on the common data from both table
--if ambiguity is there in column name(both table having the same col name) then we need to use alias support.

--this inner join returns the matching record from the DB tables based on common column.

```
> select * from dept INNER JOIN emp ON dept.did = emp.deptid;
```

Q/- give the emp details who is working in 'marketing' dept.

```
>select eid,ename,address,salary from dept INNER JOIN emp ON dept.did = emp.deptid AND dept.dname='marketing';
```

with alias support:

```
>select e.eid,e.ename,e.address,e.salary,d.dname,d.location from dept d INNER JOIN emp e ON d.did = e.deptid AND d.dname='marketing';
```

another syntax of INNER JOIN (without using INNER JOIN command).

>select e.eid,e.ename,e.address,e.salary,d.dname,d.location from dept d,emp e where d.did = e.deptid AND d.dname='marketing';

Left Outer Join:

--to get the unmatched records from the left table use left outer join (it shows the details of left table and null value for the right table).

Right Outer Join:

--to get the unmatched records from the right table use right outer join (it shows the details of right table and null value for the left table).

full outer join:

--it display the null values both side for all the unmatched records.
Note: Full Outer join is not supported by the mysql DB.

--in order to use full outer join in mysql, then we should use union of left join and right join.

```
select d.did,d.dname,e.ename,e.address from dept d LEFT JOIN emp e ON d.did = e.deptid UNION
select d.did,d.dname,e.ename,e.address from dept d RIGHT JOIN emp e ON d.did = e.deptid;
```

Self Join:
========
--here we use joining a table to itself.

--here joining condition col must belongs to same datatype.

Note:- if we want to compare two table same col value then we use INNER JOIN whereas if we want to compare 2 diff col values within a single table then we must use self join.

***whenever a table contains hirarical data then only we allow to use self join.

ex:

emp ----> manager
student ---> monitor

when we use self join, we must take the support of alias.

```
create table emp5
(
eid int primary key,
ename varchar(12),
salary int,
mgr int
);


mysql> insert into emp5 values(100,'Ram',7800,null);  // RAM does not have
any manager

mysql> insert into emp5 values(110,'Ravi',7200,100); // Ravi manager is
RAM

mysql> insert into emp5 values(112,'amit',7500,100); // amit manager is
RAM

mysql> insert into emp5 values(114,'sunil',7000,110); // sunil manager is
RAVI

Q/- display the emp name and their manager name.

--here we need to use SELF JOIN

> select e1.ename EMPLOYEE, e2.ename MANAGER from emp5 e1,emp5 e2 where
e1.mgr = e2.eid;


subqueries:
===========

--a query inside another query is called subquery or nested query.

--sub queries r used to retrieve the data from single or multiple tables
based on more than one step process.

--here outer query is called parent query and inner query is called child
query.

--child query will execute first then only parent query will be executed.

Child query :-  it provides values/data to the parent query.

Parent query : it recieves the values/data from the child query.

--in child query we can not use order by clause, but parent query can use.

--group by clause can be used in both queries.

subqueries we can categories into following categories:
---------------------------------------------------------------------

8.  single row and single col SQ(scalar value SQ)
```

9. multiple row single col SQ

10. multiple col SQ

5. single row and single col SQ(scalar value SQ):

--here child query will return only a single value.

--here mostly same col name which is in the where clause of the parent query , will be there inside the select clause of child query.

Q/- WAQ to display emp details who is working in marketing dept.

using JOIN:-

```
>select eid,ename,address from dept INNER JOIN emp ON dept.did =
emp.deptid AND dname='marketing';
```

using SQ:-

```
>select eid,ename,address from emp where deptid = (select did from dept
where dname = 'marketing');
```

Q/- WAQ to display emp details who is working with chandan;

```
> select * from emp where deptid = (select deptid from emp where
ename='chandan');
```

Q/- WAQ to display emp details who are getting more sal than avg sal from emp table.

```
> select * from emp where salary > (select avg(salary) from emp);
```

Q/ WAQ to display second higest salary employee details.

first highest salary emp:

```
> select * from emp where salary = (select max(salary) from emp);
```

second highest salary emp:-

```
>select * from emp where salary = (select max(salary) from emp where
salary < (select max(salary) from emp));
```

Q/- WAQ to display details of emp who is working under RAVI;

```
 select * from emp5 where mgr = (select eid from emp5 where ename='ravi');
```

6. Multiple row,single col SQ:

==========

--in multi-row, single col SQ, child query will return multiple rows and single col to the parent query.

--in this case in parent query we should use one of following operators:

IN
ANY
ALL

ex:
>select * from emp where salary IN (select salary from emp where eid > 100);

IN : it check equal to any number in the list (using OR)

ANY : it compare any value in the list

All : it compares all values in the list

ex: --

salary > any(----); here it checks salary should be greater than any of 4 values in the list

salary > all(----); here it checks salary should be greater than all of 4 values in the list

>any(10,20,30,40)

ex:

< any(): less than any :- less than maximum

> any(): greater than any : - greater than equal to minimum

= any(): equal to any :- it is equal to IN operator.

5 < all(10,20,30,40)

< all(): less than all :- less than minimum

> all(): greater than all : - greater than maximum

= all(): equal to all :- it is meaningless (becoz one value can not be equal to 3 or 4 value).

ex:

```
>select * from emp where salary = ANY(select salary from emp where eid > 100);
```

Q/- WAQ to display the emp who is getting max salary in each dept ?

```
>select * from emp where salary IN(select max(salary) from emp group by deptid);
```

7. multicolumn subquery:

--if we try to compare multiple col values of the child query with the multiple col values of the parent query then we use this type of SQ.

syn:

```
select * from tab_name where (col1,col2,....) IN (select col1, col2,.... from table where condition)
```

Q/- WAQ to display the emp whose salary and did matches with the salary and did of a RAM.

```
>select * from emp where (salary,deptid) IN (select salary,deptid from emp where ename='RAM');
```

SQ in DML:

SQ in insert :

```
>create table x1(id int, name varchar(12));
> insert into x1 (select eid,ename from emp);

>insert into x1 values(500, (select ename from emp where eid=100));
```

SQ in update:

here SQ is allowed inside where clause or Set clause.

ex:

```
> update x1 set name= 'ramesh' where id=(select eid from emp where ename='ram');
```

```
> update x1 set name= (select ename from emp where eid=100) where
id=(select eid from emp where ename='ram');

SQ in delete:

> delete from x1 where id = (select eid from emp where ename='ram');
```

SQL Day4

Normalization:-

--Normalization is a process of organizing the data in db to avoid the redundency(duplication).

--because of data redundency threre are several problems in the DB.

Anomalies in DB:
==========

--An anomaly is something is diff from what is normal or usual(insconsistency)
or abnormalities:

--lets try to have a single table to manage emp details:

EID	NAME	SALARY	DID	DNAME	LOCATION
12.	Ram	5000	1001	HR	DELHI
13.	Shyam	6000	1001	HR	DELHI
14.	MANOJ	6500	1001	HR	DELHI
15.	Suresh	7000	1002	SALES	MUMBAI
16.	RAVI	7200	1002	SALES	MUMBAI
17.	Ramesh	7500	1003	Accounts	Chennai

Here entire dept related data is repeated for each emp.(data redundency)

data redundency: when same data is stored multiple time unnecessarly in DB.

Redundency occurs when we try to keep all the data in a single table.

problems with data redundency:
---
11. insertion, updation, deletion anomalies
12. inconsistnecy of DB
13. increase the DB table size and slow the performance while fetching the data.

Insertion anomalies:- when certain data can not be inserted into the table without the presence of other data.

Updation anomalies: --when we want to update a single piece of data, but is must be updated at all its copies.

deletion anomalies:- if we delete some data, it cause deletion of some other data.

--to solve the above problem we need to normalize(decompose) the table into multiple related tables.

Note: the main purpose of normalization is to avoid the data redundency and maximize the efficiency of the DB.

In Normalization we should split a table in multiple tables so that each table should contains a single idea/concept.

--with the normalization we refine a big table into multiple related tables.

so the above table we should split into 2 tables:

8. dept
9. emp

--to normaize a table we have diff types of normal forms:

8. INF
9. 2NF
10. 3NF-----> upto 3NF data redundency is almost minimized or removed
11. BCNF
12. 4NF
13. 5NF

--each normal form provides a diff level of refinement of a DB.

First Normal Form:
==========

--table should not contains any multivalue attributes(comma/space seperated values)
--each cell should contains only atomic value
--a table should not have the repeating columns.

EMPID	NAME	DEPT_NAME
10.	RAM	HR, SALES
11.	RAVI	Marketing
12.	AMIT	HR, Accounts

so the above schema is in 0NF or unnormalized.

lets convert it into the 1st NF.

solution1: valid soultion

EMPID	NAME	DEPT_NAME
3.	RAM	HR
6.	RAM	SALES
7.	RAVI	Marketing
8.	AMIT	HR
9.	AMIT	Accounts

here empid can not be PK, here we need to take PK as composit PK (empid,dept_name).

solution2: invalid solution

EMPID	NAME	DEPT_NAME1	DEPT_NAME2
10.	RAM	HR	SALES
11.	RAVI	Marketing	Null
12.	AMIT	HR	Accounts

--the above solution is violates the 1NF because of repreating column.

Solution3: valid solution

--we can devide the table into 2 tables:

7. as a base table
8. as reference table

i. emp table:

EMPID	NAME
g.	RAM
h.	RAVI
i.	AMIT

j. emp_dept table

EMPID	DEPT_NAME	// here EMPID will be the FK
e.	HR	
4.	SALES	
5.	MARKETING	
6.	HR	
7.	ACCOUNTS	

2NF :
=====

To understand the 2NF or to normalize a table in 2nd NF we need to understand following concepts:

5. functional dependency
6. super key
7. candidate key

8. prime attribute
9. non-prime attribute

Key:
====
--generally we keep data in a table ,so that later we can retrieve it easy manner.

--in a table all col has a unique name but for a row we don't have a unique name. so inorder to find a row
uniquly we required a key.

--so a key is an attribute or set of multiple attributes that uniquely identify a row/record in a table.

school:----students.

ram, father_name, address, age ---->

super key:

--all the valid combination of attributes by using we can find a row uniquely in a table.

ex:

student(roll, name marks, address, dob, email)

3. roll *
4. roll name
5. roll name address
6. email
7. name email
8. name address
9. name dob
10. address email
11. dob email

Candidate key:

--it is a minimal set of super key.

Note: a candidate key should not have a subset as another super key.

1. roll

2. email

3. name address

6.name dob

Primary key:
---

here from the cadidate key DBA will choose a PK, generally it will be minimum number of attribute declared as PK.

some time we can make other than minimum value attribute also as a PK (composit key)

Prime attribute:
==

--those attribute that r part of any candidate key is called prime attribute
--and those attribute which are not part of any candidate key is known as non-prime attribute.

in the above example : marks will be a non-prime attribute.

1stNF does not eliminate redundency,

--2nF applies in a table which is having a composit key, i.e a table with a PK compound with two or more attribute.

--Note: a table with a single column PK is automatically in 2NF.

--to be in 2nF, a table must be in 1stNF and the table must not contains any partial dependency.

partial dependency:

Note: if the proper subset of a candidate key determines non-prime attribute ,then it
is called PD.

--the normalization of 1NF table to the 2nF involves the removal of PD .

--if any PD exist, we remove the partial dependecy attribute from the table by placing them in a new table.

ex:

ROLL   CID   FEE

1. c1 1000
2. c2 1500
1. c3 2000
3 c4 1000
3 c1 1000
2. c5 2000

--here there are many courses having same fee.

--here there will be only one Candidate key(roll and cid) that can uniquley indentify the record.

prime attribute:(roll and cid)
non-prime attribute (fee)

here fee is dependent on the CID, which is the example of PD. becoz CID is a proper subset of candidate key
--the above table is not in 2nF.

lets convert it into the 2NF

table 1:

(roll, cid)

table 2:
(cid, fee)

3NF:
======

Although 2ndNF relations have less redundencies than those in 1stNF still have a chance of data redundency because of transitive dependency.

--in order to make a table in 3rdNF we need to remove TD from the table.

A relation will be in 3NF if only:

1. it should be in 2NF.
2. there should not be any TD.

TD: if a non-prime attribute is transitivly dependent on primary key.
ex:

if A ------> B and B-------->C then A---->C is called TD.

ex:

Roll Name Age Country State

1. RAM 25 INDIA MP
2. RAM 35 INDIA UP
3. RAVI 28 INDIA MAHARASTRA

functional dependencies:

roll----> name
roll----> Age
roll----> State
State ----> Country

here the candidate key will be (roll)

Roll--->State and State ----> Country ,,, so Country is transitivly dependent on Roll , and it will violates 3NF,

--to convert this relation in 3NF we need to decompose this relation in multiple related tables:

Student(roll, name, age, state)

State_Country(state, country)

Transaction management:
==========

Transaction:

--it is a set of related operations used to perform a logical unit of work.

ex:

withdrawing money from the ATM
transfering amount from one account to another.

--tx access data using read and write operations.

--if a tx fails it should not be resumed, instead it should be restarted.

---->
-- 1000
-- 2000
-- ----
-- 3000
--
--
--
--
--->

500
2500

3000

in sql to manage the tx we need to use

TCL(Transaction controll language)
rollback
commit
savepoint

In order to maintain consistancy in DB , before and after the tx certain properties should be followd
these are called ACID properties:

A: (Atomicity) : (All or noting rule) the entire tx takes place at once or does not happens at all.

C: Consistancy: the DB must be consistant before and after tx

I: Isoloation: multiple tx should occurs independently at a time without any interference.

D: Durability: The changes of a sucessfull tx should be permanent even if system failure occurs.

--if we perform any DML operation on a table inside a tx area, that operation will be
partically commited(it will be stored in the local copy)

--and any time we can rollback these operation

--but once we do commit, then only it will be stored permanently.

Note: In Mysql it is bydefault autocommit is enabled.

Note: in the DB all the DDL statements are by default commited.

--in mysql to disable the autocommit mode:

>set aut0commit = 0;

to start the tx area in mysql:

>start transaction

>delete from x1 where id= 102;

>rollback;

```
>delete from x1 where id= 102;
>commit;

savepoint p1;
```