

Chapter 14

SQL

Relational database concepts:

Relational database is a collection of tables. Each table has a unique name and consists of many columns. Each column of a table also has a unique name. The name of relational database is derived from mathematical relation because there is a close correspondence between them.

Table(Relation): In RDBMS mainly data store in a special kind of object called table. In another world, a table is a collection of related entries and consists of rows and columns.

Following is an example of a student table.

Student table

Roll_no	Name	Age	Address	class
101	Harish	10	Ajmer	5th
105	Kailash	20	Kota	10th
109	Manish	18	Ahmadabad	9th
120	Ronak	14	Udaipur	8th
135	Shanker	13	Jaipur	7th

Figure1 The student relation

Field: Field of a table represents its column which is used to store the specific information about a record. For example in above given table student Roll_no, name, address and class are fields of table.

Records: It is also called the row of a table. Basically it is an individual entry of a table available in that table. For example following is the one individual entry of a student table or row of a table or tuple of a table.

Student table

105	kailash	20	kota	10th
-----	---------	----	------	------

Column: A column of a table is that vertical entry of a table which keeps all information related with a specific field. For example Roll_no is a column of a student table which keeps the following information.

Roll_no
101
105
109
120
135

Domain: Domain of a field is the set of values permitted for that field. For example domain of a field name is the set of all names.

Database schema: A schema of a database is the logical design of a database, which is rarely changed. For example following is the schema of student, classes and teacher table.

Student(Roll_no, name, age, address, class)

classes(Class_name, CStrength, CRoomNo)

Teacher(Tname, Address, salary, phone)

Relational Database instance: A collection of data or information stored in a table at any particular moment of time is called the database instance. Given below is an instance of student relation, which can be changed at any moment of time by making a new entry in the table or deleting an entry from table.

Student

RollNo	Name	Age	Address	Class
101	Harish	10	Ajmer	5 th
105	Kailash	20	Kota	10 th
109	Manish	18	Ahmadabad	9 th
120	Ronak	14	Udaipur	8 th
135	Shanker	13	Jaipur	7 th

Primary key: It is a set of one or more attributes (field) of a table which can be used to identify uniquely any row or tuples of that table. Then such set of attribute taken collectively forms a primary key of table. Primary key is also a kind of constraints. For example in above given table in figure 1 student primary key is Roll_no field because in student table using this field all the students can be identified uniquely and also with the help of Roll_no the record of any student can be easily retrieved from student table.

For example suppose the value of Roll_no is 105. If we take this value then the records retrieved from table will be of student “kailash”.

Data constraints: Data constraints are rules on columns of a table to define the limit of data entry, so that the data entries in the table should fixed the

consistency, accuracy and reliability of database and there should not be any kind of data consistency loss in database when changed is made by authorized users of database. Data constraints can be of column level and table level. The main difference between column level and table level constraints is that, column level constraints are for a single column whereas table level constraints are applied on whole table.

Following are the examples of data constraints

- 1) Class of a student can not be null.
- 2) Roll_no of two students can not be same.
- 3) There will be a matching class in classes relation for every class of student relation.

Constraints on a single relation: Following are the constraints on a single relation

- 1) Not null
- 2) Unique
- 3) Check (<predicate>)

Not null constraints: This constraints restricts the entry of null value for a field or attribute in any table that is, if this constraints is ensured for a field and any operation in this table to try to insert the value null by changing then an error will be generated.

For example, you do not want the value null for class attribute of student table. Similarly value of roll_no attribute should not be null because it is a primary key of student table.

Unique constraints: This constraint ensure that any two tuples or rows of any relation can not be same for all primary key attributes that is, for both the tuples values of all the attributes will not be same. Unique constraints form the candidate key of a relation which can be more than one in a relation.

Check constraints: Check constraints can be applied on both, domain and relation declaration. If it is applied on relation declaration then all tuples of relation must fulfill the condition specified by the check clause that is, check clause ensures that all the values of a column must fulfill the condition applied on column.

For example the values of class field attribute in student table will be 1st, 2nd, 3rd, ---, 9th, 10th, 11th, 12th.

check (class in ('1st', '2nd', '3rd', ----- '9th', '10th', '11th', '12th'))

Entity integrity constraints: Entity integrity ensures that, two rows or records of a table can not be duplicated and also the field which identifies the every records of the table is a unique field. The value of this field will not be null. Entity integrity constraints can be imposed by primary key. If we define primary key for every entity then it fulfills the entity integrity itself.

For example in given student table if the primary key of this table is Roll_no

field then Roll_no of each student in this table will be different and also the value of this field for any student will not be null that is, all will have its own Roll_no. Due to different Roll_no this table cannot have two same rows.

Student table

RollNo	Name	Address	Age	Class
110	Komal	jaipur	17	12th
120	Ronak	Udaipur	14	8th
105	kailash	kota	20	10th
107	hari	chittorgarh	10	5 th

Referential integrity: If we want to ensure that a value that appears in any relation for a given set of attributes also appears for certain set of attributes in another relation that is, both the tables have same values for some of the attributes then this condition is called referential integrity. Referential integrity can be ensured by foreign key constraints.

Foreign key integrity constraint: To understand this constraint we consider the following table as a example. There are two tables student and classes (name of tables) is given and any moment inserted values in the table is also mentioned. The primary key of given student table is Roll_no field whereas primary key of classes table is class_name field.

student

RollNo	Name	Age	Address	Class
101	Harish	10	Ajmer	5 th
105	Kailash	20	Kota	10 th
109	Manish	18	Ahmadabad	9 th
120	Ronak	14	Udaipur	8 th
135	Shanker	13	jaipur	7 th

Classes

Class_name	Class_room	Strength
12 th	F-1	95
10 th	F-2	80
9 th	F-3	70
4 th	F-4	110

Figure 2 Foreign key constraints on student relation

We are assuming that one and only one class of all standard exist such as, one class of 12th, one class of 10th, one class of 9th and so on that is sections of a class does not exist.

The student table among its attributes (Roll_no, address, age, name, class) keeps one attribute(class) which is the primary key of another table (classes).

For example class field of student table is the primary key of classes table. So this attribute class in student table is called the foreign key of student table which will refer to table classes.

The relation student is called the referencing relation whereas classes is called the referenced relation of foreign key. To become foreign key domain type of a foreign key attribute and another relation attribute should be same and number of attributes in foreign key field and another relation must be same that is, should be compatible.

We can ensure following by foreign key

- 1) We can not delete any record from classes table until a matching record is available in related table student.
- 2) We can not change the value of primary key in classes table until a related record of such record is available in another table.
- 3) We can not insert a new value in student table's class field until this value is not available in the primary key field (class_name) of classes table.
- 4) Although you can insert the null value in foreign key field but if we want that this should not reject then we use cascade in SQL.

Introduction of SQL:- SQL is a combination of relational algebra and relational calculus. It's a standard language of relational database management systems which is used to organize, manage the data available and retrieval of data from relational database. It is an important feature of SQL.

Companies like Oracle, IBM, DB2, Sybase and Ingress, use SQL as a standard programming language for their database. Basic version of this is called sequel which is developed by IBM.

Versions of SQL are SQL-86, SQL-89(extended standard), SQL-92 and SQL-1999. Latest version of SQL is SQL-2003.

SQL is not only a query language but also a standard itself which has following types:-

1. Data Definition Language (DDL)
2. Data Manipulation Language (DML)
3. Data Control Language (DCL)

Most of the commercial relational Database like IBM, Oracle, Microsoft, and Sybase etc use SQL. All Databases do not support all features available in different versions. So to use all the features of particular version always follow

manual according to the Database system.

Advantages of SQL:- SQL has many advantages for all the commercial (Oracle, IBM, DB2, Sybase) as well as for open source (MySQL, Postgres) database systems.

1. **High speed:-** SQL is a kind of language which retrieve the data efficiently and quickly from a database of a big organization. So SQL is a fast language.
2. **Easy to learn:-** It is easy to learn because of short size of programming code that is, there is no need to write many lines of code.
3. **Well defined standard:-** SQL is a standard language which is standardized by ANSI & ISO.

DDL(Data Definition Language) :- it's a part of SQL by which the schema of the database is specified. It also has specification for all the relations of database which are as follows:-

1. Specification for relational schema.
2. Specification for domain of each attribute value.
3. Specification for constraint.
4. Specification for creating index.
5. To provide authorization and security.
6. To provide physical storage for each relation.

Basic domain types of SQL:- there are few built in domain types available in all versions of SQL

1. **Numerical Data Types:-** which includes following:-

Int or Integer :- used for big size int values , occupies 4 byte of storage .

Small Int :- used for small sized integers, occupies 2 bytes of storage.

Tiny Int :- used for very small sized integers and unsigned integers.

Float(M,D):- used for floating point numbers and valid for signed numbers only. Here M is the total number of digits before decimal and D is the total number of digits after decimal

Double(M,D) only valid for floating point numbers having double precision. It is equivalent to REAL.

2. **String Type:-** which includes following:-

(i). **CHAR(C):-** this data type is valid for fixed length string. Here C is a length of the string which is given by user. If any string shorter than this size is stored then remaining space will be padded with the spaces.

(ii). **VARCHAR(C):-** used for variable sized string, here C is the maximum length of string which is specified by the user.

3. **Date or Time Date Types:-** Includes following:-

(i). **Date:-** this data type is defined in the formats YYYY-MM-DD i.e. date is

stored in this format and it can be in between 1000-01-01 to 9999-12-31. For example if age of a student in student table is 1st july 1980 then it will stored in the 1980-07-01 format.

(ii) **DATETIME** :-Date and time together can be stored in YYYY-MM-DD HH:MM:SS Formate. Date and time can be stored from 1000-01-01 00:00:00 to 9999-12-31 23:59:59. For example 2:35 PM and 1st july 1980 can be stored as 1980-07-01 14:35:00.

(iii) **Time**:- It stores time in HH:MM:SS format.

(iv) **Year**:- It stores year in 2 and 4 digit format.

Data Manipulation language (DML):- It is that part of SQL which is also known as query language so DML and query language are synonyms. DML is used to manipulate (Insert, Delete, Update and Retrieve) the data stored in any relation. The main commands of DML are:

- (i) SELECT
- (ii) UPDATE
- (iii) INSERT
- (iv) DELETE

Data Control Language :- DCL is the collection of SQL commands which provide security and maintain the rights of Data Manipulation. DCL includes the following Commands :-

- (i) COMMIT
- (ii) ROLLBACK
- (iii) GRANT
- (iv) REVOKE

DDL Commands and Syntax:-

CREATE :- CREATE Table command is used to create a table and a relation . The syntax of command is defined as following:-

Syntax: MYSQL generic syntax for creating a table is.

CREATE TABLE table_Name (F1 D1, F2 D2,..... ,Fn Dn<Integrity Constraints1,.....<ICk>);

Here in above syntax each Fi is a name of a table field or attribute and Di is the domain type of values under each Fi.

Apart from that many types of constraints can also be applied on the table like PRIMARY KEY, FOREIGN KEY etc.

For example if table named as student is to be created whose schema is defined as Student(Roll_No, Name, Age, Address, Class) and schema of Relation classes is defined as Classes(Class_Name , CRoomNo, Cstrength) then creation of student table can be defined as:

```
CREATE TABLE Student
(Roll_No int NOT NULL AUTO_INCREMENT, Name CHAR(20), Age int
NOT NULL, Address VARCHAR(30), Class CHAR(10) NOT NULL,
primary key(Roll_no), foreign key(class) references classes(class_name));
```

Here, in above example Roll_No can be defined as primary key and foreign key constraint is used to prevent the operation that try to violate the link between student and classes table. class can be taken as foreign key. Here NOT NULL and Auto increment are constraint used to define limitations i.e. NOT NULL is used to define that there must not be any NULL value in the attribute Roll_No. There would be an error if NULL value is inserted in Roll_No whereas Auto_increment is used to increase the value of Roll_no field by 1. It has value 1 by default. If we want to start the sequence by another value then we use the syntax.

```
ALTER table student AUTO_INCREMENT=100
```

Syntax for Classes table

```
CREATE Table Classes (Class_Name CHAR(10) NOT-NULL, CRoomNo
CHAR(10),
PRIMARY KEY (Class_Name));
```

Above table can be created by MySQL Prompt like

```
root@host# MYSQL-u root -p
```

```
enter password : *****
```

```
MySQL> use School_Management
```

Use command is used to use database named as School_Managemnt.

```
MySQL> CREATE DATABASE School_Management;
```

```
MySQL> use School_Management
```

```
MySQL> CREATE TABLE Student (Roll_No Int NOT NULL
AUTO_INCREMENT, Name CHAR(20), Age int NOT NULL, Address
VARCHAR(30), Class CHAR(10) NOT NULL, PRIMARY KEY (Roll_No),
FOREIGN KEY (Class) REFERENCES Classes(Class_Name));
```

-> Query ok, 0 row affected

In MYSQL termination is done by placing semicolon(;) at the end of any command or statement Any table can be removed from the database by the following command.

Generic syntax:

```
DROP TABLE table_name;
```

To remove the table named as Student the command is

```
DROP TABLE Student;
```

ALTER table command :-This command is used to add, delete, and modify the column in the database.

Following are the appropriate syntax:-

1. To add a new column

ALTER TABLE table_name ADD column_name datatype;

For example a new column Cstrength can be added to the table Classes

ALTER TABLE Classes ADD Cstrength int;

2. To remove existing column

ALTER TABLE table_name DROP COLUMN column_name;

3. to change the datatype of any existing column

ALTER TABLE table_name MODIFY column_name datatype ;

For example:-

ALTER TABLE Student MODIFY Age Date;

Other example:-

CREATE TABLE Teacher(Tname VARCHAR(20); DOB Date, Salary FLOAT(5,2),Address VARCHAR(30),phone int PRIMARY KEY (Tname));

CREATE TABLE Teaches(Tname VARCHAR(20), Class_name CHAR(10), PRIMARY KEY(Tname, Class_name), FOREIGN KEY(Tname) REFERENCES Teacher (T n a m e) , F O R E I G N KEY(Class_Name)REFERENCES Classes (Classs_Name));

Check Constraints Syntax:-

Example of Check Constraints Syntax is following

CREATE TABLE Student (Roll_No int NOT NULL AUTO_INCREMENT, Name CHAR(20), Age int NOT NULL, Address VARCHAR(30), Class CHAR(10) NOT NULL, PRIMARY KEY(Roll_No), Check(Class in ('1st', '2nd', '3rd', '4th', '5th', '6th', '7th', '8th', '9th', '10th', '11th', '12th')));

Create Index Command:-This command is used to create index of any table. Index is not visible to the user but help them to search the data rapidly in the table by the following syntax.

CREATE INDEX Index_name ON table_name (column_name)

For example

CREATE INDEX SIndex on Student(Address)

Data Manipulation commands and their syntax:-

SQL DML commands are as follows:

- (i) INSERT
- (ii) DELETE
- (iii) UPDATE
- (iv) SELECT

(i) INSERT Command:- A blank table is created by CREATE TABLE COMMAND i.e. there is no value, records and tuples in that table. To fill up the values or data in that table insert command is used.

MySQL Syntax :-

INSERT INTO
Table_Name(Column_Name1,Column_Name2,.....,Column_Namen)VAL
UES(Value1,value2,.....valuen);
To insert string values, values are enclosed within single (' ') or Double(“
”)quotes.

Insertion Through MySQL Command Prompt:-

For example if new values are to be inserted in Student table then

MySQL> USE School_Management;
MySQL>INSERT INTO Student (Name, Age, Address, Class)'
VALUES(“HARI”,15,”Chittorgarh”,”10th”);

In the above example we have not taken the column Roll_No because it is in auto increment mode which means MySQL automatically gives the value of Roll_No in sorted manner.

Other syntax is :

INSERT INTO Student VALUES (”prakash”,18,”jaipur”,”12th”);
In other syntax of INSERT INTO we can directly insert so many tuples by using SELECT instead of specifying tuples individually.

(ii) DELETE command :- An entire tuple is deleted by this command but specific value of any attribute can't be deleted by this.

Syntax:-

DELETE FROM T, WHERE P;
Here T is a relation from which tuple is to be deleted and P is a predicate (condition), according to which tuples would be deleted.
DELETE FROM Student, WHERE Roll_No=105;
By above command all the tuple having Roll_NO=105 would be deleted from the Student table. To delete all the tuples of any table we write.
DELETE FROM Student;
DELETE FROM Classes;
To delete all the records of Class 10th students from the table Classes
DELETE FROM Student, WHERE Class=”10th”;

(iii) **UPDATE command:-** UPDATE command is used to change the value of any attribute from the given table, i.e. if we do not wish to change the entire tuple but only a specific value of any attribute then update command is used.

Syntax:

UPDATE table_name SET first_field=value1, second_field=value2
[WHERE clause];

Two or more values of fields can be updated in following manner

MySQL> UPDATE Student,

SET Class="11th", WHERE Roll_No=12;

In above example, student whose Roll_No is 12th then we are try to change the class from 10th to 11th.

To change the address of a student:-

UPDATE Student , SET Address="Ajmer", WHERE Roll_No=102;
student

RollNo	Name	Age	Address	Class
11	Hari	15	Jaipur	10th
101	Prakash	16	Kota	11 th
102	Basu	11	Udaipur	6 th
120	Viveka	9	Jaipur	4 th

After update the following changes will occur in the table i.e. table will be in following form:-

Roll_No	Name	Age	Address	Class
11	Hari	15	Jaipur	10th
101	Prakash	16	Kota	11 th
102	Basu	11	Ajmer	6 th
120	Viveka	9	Jaipur	4 th

(iv) **SELECT statement:-** There are following three clauses in any SQL query expression:-

- (i) SELECT
- (ii) FROM
- (iii) WHERE

i.e. any SQL query expression consists of three clauses in its basic structure.

- SELECT clause is used to display the attributes in the output relation

- FROM clause is used to address the relation which is used in query expression. Relation written in the FROM clause join the form of Cartesian product.
- WHERE clause is used to write the predicate (condition) which is basically applied to the relation written in FROM clause and Boolean values of which (true or false)

SQL query is in this form:-

SELECT AT1, AT2, AT3, ..., ATn,
FROM r1, r2, r3, ..., rn,
WHERE P;

Here AT_i denotes attribute and r_i denote relation and P is a predicate

SELECT clause :- It keeps all the attributes which has to be retrieved from the relation.

Example :- To make it crystal clear we take the example of student table from School_Management database as followed:-

Roll_No	Name	Age	Address	Class
101	Bhagat singh	20	Ajmer	12th
105	Chandra	17	Kota	11 th
12	Shekhar	16	Jaipur	10th
17	DinDayal	15	Udaipur	9 th
90	Rohit	9	Ajmer	5th

Syntax of SELECT command is:

SELECT field_names, FROM relation_names;

Several fields can be fetched by writing field_names. We can also use astrick(*) to display all the relation in the output relation.

For example:-

SELECT Roll_No, Age, FROM Student;

Output->

RollNo	Age
101	20
105	17
12	16
17	15
90	9

SELECT * FROM Student ;

Output->

RollNo	Name	Age	Address	Class
101	Bhagat	20	Ajmer	12 th
105	Chandra	17	Kota	11 th
12	Shekhar	16	Jaipur	10 th
17	DeenDayal	15	Udaipur	9 th
90	Rohit	9	Ajmer	5 th

Duplicate fields are removed by DISTINCT keyword when used with SELECT.

SELECT DISTINCT Address FROM Student ;

The output of above query produces a relation with no duplicate Address.

Output->

Address
Ajmer
Kota
Jaipur
Udaipur

Arithmetic operations can be used in SELECT command along with field names .Operations which are used are as follows:-

Description	Operator
Addition	+
Subtraction	-
Division	/
Multiplication	*

To understand this we take a table records of all the teachers. The name of table is Teacher which contain attributes Teacher (Tname, Address, Salary, PhoneNo);

Tname	Address	Salary	Phoneno
Radha krishan	Kerla	5000	1234567899
Lalaji	Udaipur	750	1111162123
Sarvopalli	Rampur	2000	1312171080
Headgevar	Maharastra	9000	1111110510

SELECT Tname, Salary *10, FROM Teacher;

Output of above query is a relation which contain Tname , Salary, but all the

values of Salary is multiplied by 10

Output ->

Tname	Salary* 10
Radha krishnan	50,000
Lalaji	7500
Sarvopalli	20,000
Leadgevan	90,000

Use of SELECT statement with WHERE clause:-

One or more condition are written in WHERE clause and retrieval can be done only if condition is satisfied.

For example:- query for table teacher is

SELECT Tname, Salary, FROM Teacher WHERE Salary>2000;

Output ->

Tname	Salary
Radha krishnan	5000
Harivansh	9000

Following logical connections are used in WHERE clause :-

(i) AND

(ii) OR

(iii) NOT

Comparative operators are used in WHERE clause

Description	Operators
Less than	<
Less than or equal to	<=
Greater than or equal to	>=
Greater than	>
Equal to	=
Not equal to	!= or <>

Example:-

SELECT Tname, FROM Teacher, WHERE Salary>2000 AND Address="Kerala";

Output ->

Tname
Radha krishnan

3. Data Control Language commands (DCL):- These commands are related to database security. It gives privileges to users, so that users can access the some database objects.

GRANT command:- This command is used to grant permission by one user to another user to use its created objects, i.e. until or unless say user A can't grant access of its created table to user B, user B can't access it. This permission is given by using GRANT command.

GRANT statement provides many kinds of privileges on many objects (Table, View)

Syntax:-

GRANT [type of permission] ON [database name] .
[table_name] TO '[user name]' '@'localhost';

Here types of permissions are.

- CREATE- Give permission to create new table or database.
- DROP - Give permission to drop the table or database.
- DELETE - Give permission to delete lines or tuples.
- INSERT- Give permission to insert a new lines or tuples.
- SELECT - Give permission to read the table or database.
- UPDATE- Give permission to update the tuples.

If we use asterisk (*) in place of database name or table name in above syntax then it gives the permission to access any database or any table.

Syntax:-

GRANT ALL Privileges ON *.* TO 'new_user' '@'localhost';

This command gives permission to read, write ,execute and to perform all the operations on all the databases and tables.

Example- To understand GRANT command in MySQL

Create New Users:- Default user in MySQL is Root, which has full access on all the databases .To create new user the syntax is

MySQL> CREATE USER 'new_user' '@'localhost' IDENTIFIED BY 'Password';

Example:-

MySQL> CREATE USER '14EEACS350' '@'localhost' IDENTIFIED BY '123456';

Using above syntax a new user named as '14EEACS350' is created and password is '123456' , although this user doesn't have any kind of access to database even though this user is not allowed to login therefore privileges are to be given to the user.

Example:- GRANT ALL PRIVILEGES ON
School_Management.* TO '14EEACS350'@'localhost'
IDENTIFIED BY '123456';

By above example user 14EEACS350 is given access on all the tables of School_Management database.

Execute following command once access is given to the user.

FLUSH PRIVILEGES;

To make all the changes effective

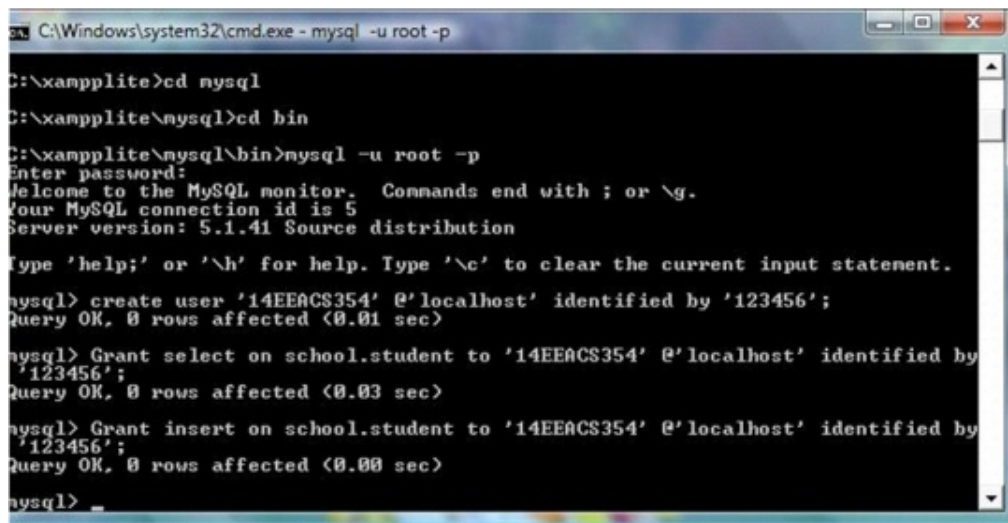
If administration doesn't want to give all the privileges to user '14EEACS350', then to give access to read only,

GRANT SELECT ON School_Management.Student TO '14EEACS350' @
'localhost' IDENTIFIED BY '123456';

To give privileges of insertion in Student table

GRANT INSERT ON School_Management.Student TO '14EEACS350' @
'localhost' IDENTIFIED BY '123456';

Like above example, other privileges can be given to the users. Below is the screen shot for assigning privileges insert and select to new user.



```
C:\Windows\system32\cmd.exe - mysql -u root -p

C:\xampplite>cd mysql
C:\xampplite\mysql>cd bin
C:\xampplite\mysql\bin>mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 5
Server version: 5.1.41 Source distribution

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> create user '14EEACS354' @'localhost' identified by '123456';
Query OK, 0 rows affected (0.01 sec)

mysql> Grant select on school.student to '14EEACS354' @'localhost' identified by
'123456';
Query OK, 0 rows affected (0.03 sec)

mysql> Grant insert on school.student to '14EEACS354' @'localhost' identified by
'123456';
Query OK, 0 rows affected (0.00 sec)

mysql> _
```

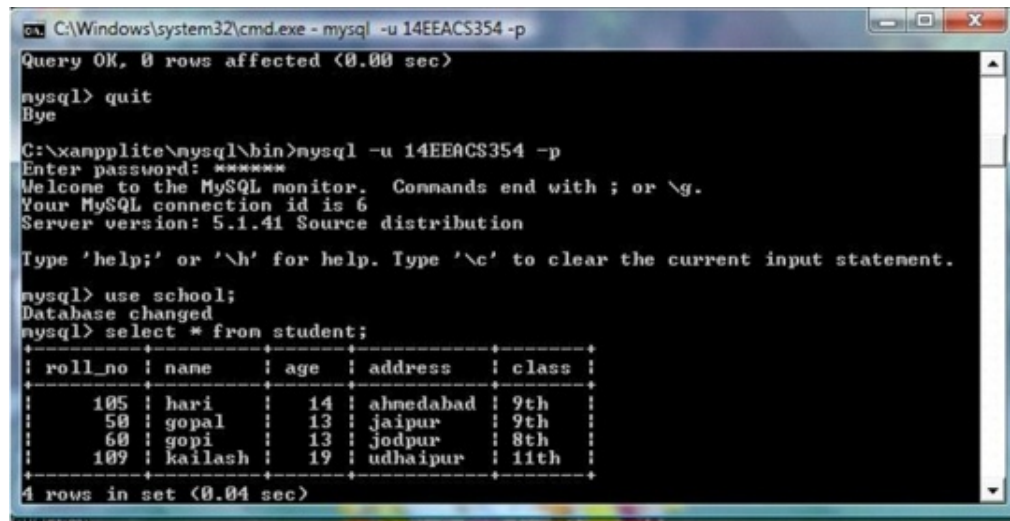
Figure 3 Screen shot of privileges assigning to new user

User '14EEACS350' is only given privileges to insert and select. So this user is allowed to perform select and insert on student table and not any other operation.

To see this we first logout by entering QUIT command and then again go for new login

MySQL-u[new username]-P;

MySQL> MySQL-u 14EEACS350 -P;
Enter password:123456;
And then perform different command on Student table following are some screen shots.



```
C:\Windows\system32\cmd.exe - mysql -u 14EEACS354 -p
Query OK, 0 rows affected (0.00 sec)

mysql> quit
Bye

C:\xanpplite\nmysql\bin>mysql -u 14EEACS354 -p
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 6
Server version: 5.1.41 Source distribution

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use school;
Database changed
mysql> select * from student;
+-----+-----+-----+-----+-----+
| roll_no | name  | age  | address | class |
+-----+-----+-----+-----+-----+
| 105     | hari  | 14   | ahmedabad | 9th   |
| 50      | gopal | 13   | jaipur   | 9th   |
| 60      | gopi  | 13   | jodpur   | 8th   |
| 109     | kailash | 19   | udhaipur | 11th  |
+-----+-----+-----+-----+-----+
4 rows in set (0.04 sec)
```

```

C:\Windows\system32\cmd.exe - mysql -u 14EEACS354 -p
'123456';
Query OK, 0 rows affected (0.00 sec)

mysql> quit
Bye

C:\xampp\mysql\bin>mysql -u 14EEACS354 -p
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 6
Server version: 5.1.41 Source distribution

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use school;
Database changed
mysql> select * from student;
+-----+-----+-----+-----+-----+
| roll_no | name  | age | address | class |
+-----+-----+-----+-----+-----+
| 105     | hari  | 14  | ahmedabad | 9th   |
| 50      | gopal | 13  | jaipur   | 9th   |
| 60      | gopi  | 13  | jodpur   | 8th   |
| 109     | kailash | 19  | udhaipur | 11th  |
+-----+-----+-----+-----+-----+
4 rows in set (0.04 sec)

mysql> insert into student values(54, 'board', 10, 'ajmer', '5th');
Query OK, 1 row affected (0.03 sec)

mysql> select * from student;
+-----+-----+-----+-----+-----+
| roll_no | name  | age | address | class |
+-----+-----+-----+-----+-----+
| 105     | hari  | 14  | ahmedabad | 9th   |
| 50      | gopal | 13  | jaipur   | 9th   |
| 60      | gopi  | 13  | jodpur   | 8th   |
| 109     | kailash | 19  | udhaipur | 11th  |
| 54      | board | 10  | ajmer    | 5th   |
+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)

mysql> delete from student where roll_no=60;
ERROR 1142 (42000): DELETE command denied to user '14EEACS354'@'localhost' for table 'student'
mysql> _

```

REVOKE command :- This command is used to take back the privileges from any object(table) and its syntaxes are same as to GRANT command.

Syntax:- REVOKE [type of permission] ON [database_name].
[table_name] FROM'[user name] '@'localhost';

For example :- DROP can be used to eliminate any user.

DROP USER '14EEACS350' '@'localhost';

Example:-

REVOKE DELETE ON Student FROM 14EEACS350;

COMMIT command :- It is used to make the changes permanent in the database i.e. if any change apply to the database is committed by commit

command. By default auto commit mode is enabled in MySQL i.e. if any changes or updating of a table occurs, then that updating is stored on MySQL disk to make it permanent. We can disable auto commit by START TRANSACTION.

ROLLBACK:- the rollback undo all the transaction to a certain checkpoint before failure has occurred. In case of cascaded rollback there are multiple failure due to single failure which causes multiple times of rollback in different transaction.

Introduction to SQL operators :- These are reserved words which are used in WHERE clause of any SQL query. Main operators are :-

- (i) Comparison
- (ii) Arithmetic
- (iii) Logical
- (iv) Operators used to negate the condition.

SQL comparison operators:-

Operators	Description
(=)	this operator is used to check the equality or non equality of two operators. Condition is true in case of equality.
(<> or !=)	this operator is used to check the equality or non equality of two operators. Condition is true in case of non equality.
(>)	if the value of left hand side operator is greater than right hand side operator then condition is true.
(<)	if the value of left hand side operator is less than right hand side operator then condition is true.
(>=)	if the value of left hand side operator is greater than or equals to right hand side operator then condition is true.
(<=)	if the value of left hand side operator is less than or equals to right hand side operator then condition is true.

SQL Arithmetic operators:-

Operators	Descriptions
(+)	used to add left hand side operand and right hand side operand.
(-)	used to subtract the value of right hand side operand from left hand side operand.
(*)	used to multiply left hand side operand and right hand side operand.
(% or modulus)	Right hand side operand divides left hand side operand

and return remainder as a result.

To understand arithmetic and comparison operators consider the following student, teacher and classes table as example:-

Student

Roll No	Name	Age	Address	Class
109	Omprakash	9	Jodhpur	4th
111	Prakash	15	Jaipur	10th
91	Suman	11	Jaipur	6th
75	Shanker	13	Ajmer	8th

Teacher

Tname	Address	Salary	PhoneNo
Radha krishnan	Kerla	3000	1234567899
Rajesh	Jodhpur	5000	9413962123
Lalaji	Ajmer	9000	9312171080
Hariom	Kerla	40000	5189310510

Classes

Class_name	CroomNo	CStrength
12th	F-1	90
6th	F-17	65
9th	S-21	110

Figure 4 Student , Teacher and classes table

Example 1.

```
SELECT * FROM Student, WHERE Age=15;
```

Output ->

Roll_No	Name	Age	Address	Class
111	Prakash	15	Jaipur	10 th

Because there is only one row with Age=15 in student table

Example 2.

```
SELECT * FROM Student, WHERE Age>11;
```

Output ->

Roll_No	Name	Age	Address	Class
111	Prakash	15	Jaipur	10 th
75	Shanker	13	Ajmer	8th

Example 3.

SELECT * FROM Student, WHERE Age <= 13;

Output ->

Roll_No	Name	Age	Address	Class
109	Omprakash	9	Jodhpur	4th
91	Suman	11	Jaipur	6th
75	Shanker	13	Ajmer	8th

SQL Logical operators:-

Operators

Description

AND a AND b is true if both(a and b) of them is true.

OR a OR b is true if either a or b is true.

For example:-

Consider a table Teacher

(i) SELECT Tname , Salary FROM Teacher WHERE Salary <= 5000 and Salary >=4000;

Output ->

Tname	Salary
Rajesh	5000

Here in this example only one record is available where both the conditions, Salary <= 5000 and Salary >=4000 is satisfied.

(ii) SELECT Tname , Salary, FROM Teacher, WHERE Salary <= 5000 OR Address = "Kerla";

Output ->

Tname	Salary
Radha krishnan	3000
Rajesh	5000
Hariom	40000

Here in this example there are three entries are found as a result because both conditions Salary <=5000 and second condition address=" Kerla" are true for

two rows. So results will be found as above.

Operators **Descriptions**
 Not(!) It inverts the value of an operand.

Example

SELECT * FROM Student WHERE !(Roll_No= 111);

Output ->

RollNo	Name	Age	Address	Class
109	Omprakash	9	Jodhpur	4th
91	Suman	11	Jaipur	6th
75	Shanker	13	Ajmer	8th

Above query returns all the records excluding the records containing Roll_No =111.

Operators **Descriptions**
 BETWEEN It actually search value between two values of attribute.

Example :-

SELECT Tname , Salary FROM Teacher WHERE Salary BETWEEN 3000 and 5000;

Output ->

Tname	Salary
Radha krishnan	3000
Rajesh	5000

Operators **Descriptions**
 ANY this operator is used to compute a value to the other values in the list.

Syntax:- ANY operator can has following syntax
 <ANY, <=ANY, >=ANY, =ANY and <>ANY

Example - If say value 5 is to be compared with other values in the list

0
5
6

Then resulting relation using ANY:-

Expression	Result	Remark
(i) 5 > ANY {0,5,6}	True	because 5 is greater than 0

Output ->

Roll_No	Name	Age	Address	Class
91	Suman	11	Jaipur	6th
75	Shanker	13	Ajmer	8th

Example (2): Names ending with “sh”;

MySQL> SELECT * FROM Student WHERE Name LIKE '%sh';

Output ->

Roll_No	Name	Age	Address	Class
109	Omprakash	9	Jodhpur	4th
111	Prakash	15	Jaipur	10 th

Example 3: Find all the names consist of 5 characters

MySQL> SELECT * FROM Student WHERE Name LIKE '-----';

Output->

Roll_No	Name	Age	Address	Class
91	Suman	11	Jaipur	6th

If a special character (%,_) is to be added in some pattern then escape character (\) Backslash is to be used by placing it before the special character.

Example:-

- LIKE 'BJ\%BHARAT%' will match all the string starts with BJ%BHARAT
- LIKE 'BJ\\BHARAT%' will match all the string starts with BJ\BHARAT

SQL NOT LIKE is used to find mismatches. Extended regular expression are used for pattern matching in SQL which are of two types . REGEXP(RLIKE) and NOT REGEXP(NOT RLIKE).

“[...]” is a character class which matches any character inside the bracket.

Example “[pqr]” matches “p”, “q” or “r”

“[a-z]” matches any letter

Operators Descriptions

IS NULL this operator is used to compare any value with a NULL value .

If no value exist in any field of an attribute than NULL value is used to represent it.

Example:- if no PhoneNo exists for a teacher in teacher table then, there would be a NULL value and for testing of that key word NULL is used in Mysql

Example:- SELECT Tname FROM Teacher WHERE PhoneNo IS NULL;

Operators and NULL values :-

Operator	Value1	Value2	Output
+, -, *, or /	value	NULL	NULL

+, -, * or /	NULL	value	NULL
>, <, >=, <=, <>	NULL	value	unknown
<=, >=, <>, <=, <>, =	value	NULL	unknown
AND	true	unknown	unknown
AND	false	unknown	false
AND	unknown	unknown	unknown
OR	true	unknown	true
OR	false	unknown	unknown
OR	unknown	unknown	unknown
NOT	unknown	unknown	unknown

That is, if result of predicate in WHERE clause is false or unknown than no tuple displays in result .

NOTE:- all aggregate function except (count(*)) will ignore the NULL value as their input calculation .

SET Operators:- set operations are applied to the relations are as follows:-

- (i) UNION
- (ii) UNION ALL
- (iii) INTERSECT
- (iv) EXCEPT

UNION set operator :- This operator is used to combine the result set of two or more than two SELECT statement. It removes duplicate tuples.

NOTE :-UNION can be applied to only those relations which have same number of fields and their data type must also be same.

Syntax:- MySQL UNION operator syntax is

SELECT ex1, ex2, ... ,ex_n FROM tables [WHERE condition]

UNION [DISTINCT]

SELECT ex1, ex2, ... ex_n FROM tables WHERE condition;

Here, DISTINCT keyword is not required because UNION itself removes the duplicates.

Example:-

SELECT Class FROM Student

UNION

SELECT Class_name FROM Classes;

Output ->

Class
4th
10th
6th
8th
12th
9th

Here in this example there is only one field i.e. SELECT statement return only one field. Data types of both the fields are same. Column name of return relation will be the name of first SELECT statement column name because we know that UNION will remove all duplicates that's why value 6th occur only once in the relation. But if want to keep all duplicates tuples then we will use UNION ALL.

Syntax :- Syntax of UNION ALL is same as the syntax of UNION what matters is that UNION ALL IS written instead of UNION.

SQL INTERSECT operator:- It is used to return a relation which contains the tuples common to given relations i.e. if there exist any tuple in two or more than two relation than that common tuple will be the result of intersection of the two relations.

Example:-

```
SELECT Class FROM Student
INTERSECT
SELECT Class_name, FROM Classes;
```

Output ->

Class
6th

Note:- Intersection is not available in MySQL but can be done through IN OPERATOR.

Syntax:-

MySQL IN OPERATOR syntax

Expression IN(value1 , value2 , ...value_n);

Here, expression is value which we want to test and value1,value2,...valuen are values in which we have to test the value.If any value matches the test value then IN OPERATOR returns true.

SQL EXCEPT operator :- it combines two SELECT statement and return a relation which contain those tuples of first SELECT statement which are not in

the second SELECT statement.

Syntax:-

SELECT column names FROM tables [WHERE clause]

EXCEPT

SELECT column names FROM tables WHERE clause;

EXCEPT operator is not available in MySQL but to fulfill the needs NOT IN operator is used.

SQL functions :- Many built in functions are available in SQL which are as follows:-

(i) Date and Time function:- date and time functions and there description which are used in MySQL are as follows:-

Function name	description
ADDDATE()	to add dates
ADDTIME()	to add times
CURDATE()	return a present date
CURTIME()	return current time
DATE_SUB()	subtracts two dates
NOW()	it return present date & time
STR_TO_DATE()	it changes string into date

Example 1. Syntax:- SELECT ADDDATE(expr, days)

MySQL> SELECT ADDDATE('1980-07-01',32);

Output -> 1980-08-02

Example 2. Syntax:- SELECT ADDTIME(expr2, expr1)

Here expr1 is added to expr2

MySQL> SELECT ADDTIME('1999-12-31 23:59:59.999999',
'11:1:1.000002');

Output -> 2000-01-02 01:01:01.000001

Example 3.

CURDATE syntax :- SELECT CURDATE();

Return present date in YYYY-MM-DD format.

String function :- useful string functions of MySQL are as follows:-

Name	description
ASCII()	return a numeric value of left most character
BIN(N)	return a string representation of binary value of N
BIT_LENGTH(str)	return the length of string in bits

CHAR(N)	this function consider each argument N as an integer and gives its string represent. This string is the combination of all those characters which are passed as an argument.
CHAR_LENGTH(Str)	it measure the string length in characters.
CONCAT(Str1,Str2,...)	augmented strings are concatenated and return as result.
FIELD(Str,Str1,Str2,...)	it return an index of string str from the given list, if string is not obtained than result will be 0.
LOAD_FILE(file_name)	read the file and return its contain in the form of string. To use this path to that file must be specified.
REPLACE(Str, from_Str, to_Str)	it returns a string str after removing all the occurrences of from_Str and replace it by to_Str.

There are many functions in MySQL except the listed ones.

Example1:- MySQL> SELECT ASCII('3')

Output ->

ASCII('3')
51

Example2:- MySQL> SELECT BIN(2)

Output -> 1 0

Example3:- MySQL> SELECT BIT_LENGTH('BHARAT')

Output ->

BIT_LENGTH('BHARAT')
48

Example4:- MySQL> SELECT CONCAT('BH','A','GAT',' ','SI','NGH')

Output -> BHAGAT SINGH

Example5:- MySQL> UPDATE Student SET Address=
LOAD_FILE('pathname') WHERE Roll_No=105 ;

Example6:- MySQL> SELECT CONCAT(Roll_No, Name, Class) FROM
Student

Output ->

CONCAT(Roll_No, Name, Class)
105hari9th
50gopal9th
60gopi8th
109kailash11th

Example7:- MySQL> SELECT CHAR(66,72,65,82,65,84)

Output->

CHAR(66,72,65,82,65,84)
BHARAT

RAND function:- Is used to generate any number randomly between 0 to 1 in MySQL.

Example :- MySQL> SELECT RAND(), RAND();

Output-> RAND()

RAND()
0 . 0 3 0 1 4 5 6 7 8 4 5 3 5 7

0.93969467893221

SQRT function :- It is used to calculate the square root of any Number.

Example :- MySQL> SELECT SQRT(64)

Output ->

SQRT
8

If square root of Salary field is calculated from Teacher then this is done as follows-

MySQL>SELECT Tname, SORT(Salary)
FROM Teacher

Output->

Tname	Salary
Radha Krishnan	54.7722557505166
Rajesh	70.7106781188548
Lalaji	94.8683298050514
Hariom	200

Numeric functions:- these functions are used in mathematical operations.

Few important functions are listed.

Function	Description
(i)ABS(V)	it returns a full value of function V
(ii)GREATEST(n1,n2,...)	it return the greatest value among the parameter list values.
(iii)INTERVAL(N,n1,n2,n3,----)	it compairs the value of N to the values(n1,n2.....) one by one if N<n1it return 0 else return 1 for N<n2 and return 2 For N<n3.
(iv) LEAST(N1,N2....)	It is a inverse of GREATEST

Example(1) MySQL> SELECT ABS(-6);

Output ->

ABS(-6)
6

Example(2) MySQL> SELECT GREATEST(4,3,7,9,8,0,10,50,70,11)

Output ->

GREATEST(4,3,7,9,8,0,10,50,70,11)
70

Example(3) MySQL> SELECT INTERVAL(4,3,5,8,11,12,17,18)

Output ->

INTERVAL(4,3,5,8,11,12,17,18)
1

Aggregate functions:- This function accepts collection of values as an input in MySQL and return a single value as an output .There are five types of built in aggregate function in MySQL.

AVERAGE: Avg()
MAXIMUM: Max()
MINIMUM: Min()
TOTAL: Sum()
COUNT: Count()

Here input to the function Sum and Average must be numbers while other operators can work on string also.

Avg() function: This function is used to calculate the average value of given field values.

Example:- MySQL> SELECT Avg(Salary) FROM Teacher;

Output->

Avg(Salary)
14250.0000

In above example Avg() function return the average value of Salary field values.

Sum() Function: This function return the Sum of all the values of any fields.

Example:- MySQL> SELECT Sum(Salary) FROM Teacher;

OUTPUT->

Sum(Salary)
57000

Max() function:- return the maximum value among the value of any record set

Example :- MySQL> SELECT Max(Salary) FROM Teacher

Output->

Max(Salary)
40000

Min() function:- It returns record with minimum value.

Example :- MySQL> SELECT Min(Salary) FROM Teacher

Output->

Min(Salary)
3000

Count() function :- it is used to count the number of records in the table i.e. total number of records can be calculated.

Examplem1:- SELECT Count (*) FROM Student

Output ->

Count(*)
4

Example2 :- SELECT Count (*) FROM Student WHERE Class="9th"

Output ->

Count(*)
2

SQL ORDER BY clause:- to sort the rows in proper order "ORDER BY" clause is used or by this clause we can sort any columns value either in ascending or descending order.

Syntax:-

SELECT field1,field2....fieldn FROM T1,T2...Tn ORDER BY field1,

field2....fieldn [Asc[Desc]];

Clause can be applied to many number of fields for which keyword Asc and Desc is used .Asc stand for ascending and Desc stands for descending.

Example 1:- MySQL> SELECT Roll_NO, Age FROM Student ORDER BY Age Desc;

Output:-

Roll_No	Age
109	19
105	14
50	13
60	13

Example 2:- MySQL> SELECT Roll_NO, Age FROM Student ORDER BY Age Desc, Roll_No Desc;

Output:-

RollNo	Age
109	19
105	14
60	13
50	13

Example:- Query :- find all the students who sits in room number F-17

MySQL> SELECT Name FROM Student, Classes WHERE Class=Class_name AND CRoomNo="F-17" ORDER BY Name Asc;

In this example many tables are used because we required information from different tables. Like student name can be retrieve from Student table and CRoomNo field can be retrieved from Classes table. Both the tables are joined by their primary and foreign key. Because both the keys are same for the table i.e. tables are linked through their keys. Two tables are joined by a field which is common to the table. Here ORDER BY clause is used because desired result must be in increasing order.

Output->Students who want to sit in F-17

Name
--

SQL GROUP BY clause:- It is useful clause of MySQL. Many important

queries can be written by this clause. Collection of values of one or many columns can be grouped by this clause that is attributes mentioned in group by clause are used to form groups. Attributes or attributes values which are same for all tuples will represent a single group.

This clause can be understood by following example of student table:

Roll_No	Name	Age	Address	Class
1	Ajay	9	Jaipur	4th
2	Vijay	17	Kota	12th
10	Hari	11	Udaipur	7th
17	Shanker	13	Jaipur	8th
21	Om	21	Ajmer	12th
51	Mayank	15	Ajmer	9th
90	Anju	18	Ajmer	11th
53	Suman	12	Ajmer	10th
64	Kamal	10	Kota	4th
500	Komal	16	Udaipur	9th
700	Aryabhatt	11	Jaipur	7th
900	Bodhayan	13	Jodhpur	8th

Figure 5 Student table

Example:- Query : find the number of student in each class

If we write the syntax like that then result obtained will be wrong.

MySQL> SELECT Count(*) FROM Student

Output->

Count (*)
12

By above query result will show the total number of student i.e. total number of students in the table is same as the total number of student in the school. Therefore it returns the total number of tuples in the relation. For appropriate result GROUP BY clause is used with Count aggregate function and syntax is MySQL> SELECT Class, Count(Roll_No) FROM Student GROUP BY Class;

After grouping Student table can be depicted as given below because group is making by Class attribute in GROUP BY clause so tuples with same Class will be display in a group.

Output ->

Class	Roll_No	Age	Name	Address
11th	90	18	Anju	Ajmer
12th	2	17	Vijay	Kota
12th	21	21	Om	Ajmer
10th	53	12	Suman	Ajmer
9th	51	15	Mayank	Ajmer
9th	500	16	Komal	Udaipur
8th	17	13	Shanker	Jaipur
8th	900	13	Bhodhayan	Jodhpur
7th	10	11	Hari	Udaipur
7th	700	11	AryaBhatta	Jaipur
4th	1	9	Ajay	Jaipur
4th	64	10	Komal	Kota

Class	Count(Roll_No)
10th	1
11th	1
12th	2
9th	2
8th	2
7th	2
4th	2

The column by which group is made , any calculation like Count, Avg, Max, Min etc can be applied by aggregate function. Therefore in this query Count aggregate function can be applied for each groups tuples having same class value because there are only two fields in SELECT statement so following result are obtained.

Class	Count(Roll_No)
10th	1
11th	1
12th	2
9th	2
8th	2
7th	2
4th	2

Note :- any attribute which occur outside the aggregate function in SELECT

statement are written inside the GROUP BY clause.

For example-> Class attribute which appears in SELECT statement will also be in GROUP BY clause.

Example 2:- Name all the classes in which number of student are more than 1.

Syntax:-

MySQL> SELECT Class, Count(Roll_No) FROM Student GROUP BY Class
HAVING Count(Roll_No);

Output->

Class	Count(Roll_No)
12th	2
9th	2
8th	2
7th	2
4th	2

Above output contains only those tuples whose count is more than 1. To check condition for a single tuple in SQL can be done by using WHERE clause But to see the condition for tuples in the groups made by GROUP BY clause, HAVING clause is used. This is the main difference between WHERE and HAVING clause.

Predicate of HAVING clause are applied after making the group by GROUP clause. Therefore aggregate function can also be used with it.

Note:- If WHERE, HAVING, GROUP BY occur simultaneously then predicate is applied to WHERE clause first after that all the tuples for which condition is satisfied will be kept in the same group by group clause. At the end having clause is applied for each group. Groups which don't fulfill the condition of having will be extracted.

SQL RENAME operation:- Name of any relation and attributes can be changed by RENAME operation using 'AS' clause. It is used for renaming and syntax is.

Old_relation/ attribute_name as new_name 'AS' clause can be used in both SELECT and FROM statement.

Example1:- MySQL> SELECT Avg(Salary) AS AvSalary FROM Teacher;

AvSalary
14250.0000

Example 2:- MySQL> SELECT Class, Count(Roll_No) AS total, FROM Student GROUP BY Class;

Output->

Class	total
10th	1
11th	1
12th	2
9th	2
8th	2
7th	2
4th	2

SQL JOIN:- JOIN keyword is used to combine two or more than two tuples from the given relation. In joins two relations are taken as input and return a single relation as an output.

There are many mechanisms to perform joins:-

- (1) Cartesian product mechanism
- (2) Inner join
- (3) Outer join (left, right, full)

Every join type stated earlier also has one Join condition with it. So each Join expression is consists of a Join type and a Join condition which is used in FROM clause.

To understand the operation of Join we consider two tables, Student given in figure 5 and Classes table of figure 6

Class_name	CRoomNo	CStrength
12th	F-1	90
11th	F-2	75
10th	F-5	99
9th	S-21	110
8th	S-10	70
7th	F-10	85
6th	F-17	65
5th	F-7	60
4th	F-9	55
3th	F-8	50
2th	S-15	35
1th	S-9	60

Figure 6 Classes table instance

We write a query to join the Student and Classes table

```
MySQL> SELECT Roll_No, Class, CStrength FROM Student AS St, Classes
AS S, WHERE St.Class=S.Class_name;
```

Here Student is renamed as St and Classes as S. Cartesian product is applied to both the relation where each tuple of St is joined with each tuple of S. So total number of tuples in resultant relation is.

$$N1 * N2 = 12 * 12 = 144$$

Here N1 is total number of tuples in St table and N2 is the total number of tuples in S table but result relation contains only those tuples which fulfils the condition of WHERE clause.

OUTER JOIN OPERATIONS:- These are of following types.

- LEFT OUTER JOIN
- RIGHT OUTER JOIN
- FULL OUTER JOIN

Following join condition are used with Outer Joins:-

- 1) Natural
- 2) ON (Predicate)
- 3) Using (A1,A2,.....An)

Left outer join and ON Join condition:- Two Understand this consider two tables

Classes

Class_name	CRoomNo	CStrength
12th	F-11	90
9th	S-21	110
7th	F-10	85
4th	F-9	55

Admission

Class_name	Roll_No	admission_date
12th	79	2000/7/15
9th	89	2010/8/13
6th	69	2012/7/21
5th	49	2013/7/03

Figure 7 Classes and Admission relation

Syntax:-

Select Classes, Class_Name, Roll_No, CStrength From Classes Left OuterJ
oin Admission on Classes.class_Name=Admission.Class_Name

Here Name of Relation is written with it's attribute because Name of attributes

is common in both the relation so to avoid the ambiguity we have written name of relation along with attribute name.

Output

Class name	Roll No	CStrength
12th	79	90
9th	89	110
7th	Null	85
4th	Null	55

Matching tuples of both the relation and unmatched tuples of left side relation are present in the result relation.

RIGHT OUTER JOIN AND ON JOIN CONDITION:-

Syntax:

Classes Right Outer Join admission on
classes.Class_Name=Admission.Class_Name

Output

Class_ Name	CRoom No	CStength	Class_ Name	Roll_ No	Admission_ date
12th	F-11	90	12th	79	2000/7/15
9th	S-12	110	9th	89	2010/8/13
Null	Null	Null	6th	69	2012/7/21
Null	Null	Null	5th	49	2013/7/03

Right outer Join is as same as left outer join but the only difference is that unmatched tuples of R.H.S relation of Join operation will be present in the result relation. Null value is assigned for the attributes of left relation.

Full-outer join ON condition:-

Syntax:

Classes full outer join admission on classes. Class_Name = Admission.
Class_Name

Class_ Name	CRoom No	CStrength	Class_ Name	Roll_ No	Admission_ date
12th	F-11	90	12th	79	2000/7/15
9th	S-12	110	9th	89	2010/8/13
7th	F-10	85	Null	Null	Null
4th	F-9	55	Null	Null	Null
Null	Null	Null	6th	69	2012/7/21
Null	Null	Null	5th	49	2013/7/03

Here in result unmatched tuples of both the relation occur and Null Values for unmatched tuples of other relation is assigned.

Outer Join and Natural condition:- When we perform natural join of two relation, the number of tuples contain in result relation depends on the common attributes in both the relation. The common attributes appears first in result relation with single copy(without duplicate).

Example

Classes Natural right outer join admission

Class_ Name	CRoom No	CStength	Roll_ No	Admission_ date
12th	F-11	90	79	2000/7/15
9th	S-12	110	89	2010/8/13
6th	Null	Null	69	2012/7/21
5th	Null	Null	49	2013/7/03

Other outer joins for natural join condition can also be obtained like above

Inner join :- Example-

Classes inner Join Admission On Classes.Class_Name=
Admission.Class_Name;

Output->

Class_ Name	CRoom No	CStength	Class_ Name	Roll_ No	Admission_ date
12th	F-11	90	12th	79	2000/7/15
9th	S-12	110	9th	89	2010/8/13

Inner join and natural condition:-

Example :- Classes Natural Inner-Join Admission

Output ->

Class Name	CRoomNo	CStength	Roll No	Admission date
12th	F-11	90	79	2000/7/15
9th	S-12	110	89	2010/8/13

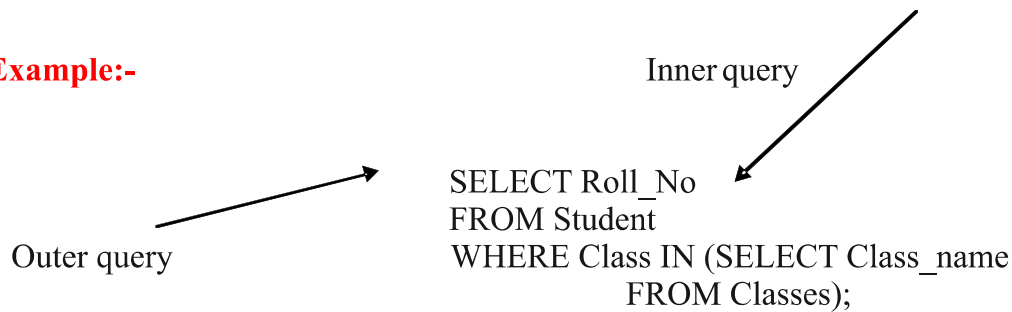
Here only one Attribute is common to both the relation, therefore Join is performed by that attribute only.

Note:- Join condition USING is also same as Natural Join only joining attributes in USING will be (A1,A2,...An) rather than all common attribute present in both relation which is the basic difference between using and natural join condition.

SQL Sub queries:- A sub query is a SQL query which is nested within another

query. A sub query can also be nested within another sub query. Sub query is called the inner query and the query in which a sub query is nested is called the outer query.

Example:-



The value return by a sub query can also be compared using comparison operators (=,>,< etc).

For example :- To understand this we use the Teacher table
 SELECT Tname, Salary FROM Teacher WHERE Salary= (SELECT
 Max(Salary) FROM Teacher);
 Output ->

Tname	Salary
Hariom	40000

Example 2:- Query- Find the name of teacher having salary, less than the average salary of all teacher
 SELECT Tname, Salary FROM Teacher WHERE Salary< (SELECT
 Avg(Salary) FROM Teacher);
 Output ->

Tname	Salary
Radha krishnan	3000
Rajesh	5000
Lalaji	9000

Important Points:

- A schema of a database is the logical design of a database , which is rarely changed.
- A collection of data or information stored in a table at any particular moment of time is called the database instance.
- **Primary key** :It is a set of one or more attributes (field) of a table which

can be used to identify uniquely any row or tuples of that table.

- Referential integrity can be ensured by foreign key constraints.
- SQL is a combination of relational algebra and relational calculus.
- Auto_increment is used to increase the value of a field by 1.
- GRANT ALL Privileges ON *.* TO 'new_user @ 'localhost'; This command gives permission to read, write ,execute and to perform all the operations on all the databases and tables.
- Attributes mentioned in group by clause are used to form groups. Attributes or attributes values which are same for all tuples will represent a single group.
- There are many mechanisms to perform joins e.g. Cartesian product mechanism, Inner join and Outer join (left, right, full).

Practice Questions

Objective type questions:

- Q1. Which one is not a SQL clause?
- a) Select
 - b) From
 - c) where
 - d) condition
- Q2. Full form of SQL is
- a) Structure Question language
 - b) syntax question language
 - c) Structure query language
 - d) Structure question language
- Q3. DDL stands for.
- a) Data definition language
 - b) Dual data language
 - c) Data data language
 - d) none of these
- Q4. Count() is a
- a) String function
 - b) numeric function
 - c) both
 - d) not exist
- Q5. Like operator is used for
- a) Concatenating strings
 - b) count string character
 - c) string matching
 - d) all

Very Short answer type questions.

- Q1. What is SQL?
- Q2. What do you understand by SQL from clause?

- Q3. What is the importance of SQL select clause?
Q4. Give names of types of SQL.
Q5. What is the difference between unique and primary constraints?
Q6. Define database instances.
Q7. How we use order by clause in SQL.
Q8. What is NULL in SQL?
Q9. What do you mean by aggregate functions?
Q10. Why we use SQL grant command.

Short answer type questions:

- Q1. What is the basic structure of SQL?
Q2. What are the various DML commands? Give syntaxes for them.
Q3. What is foreign key? How we create a foreign key in a table?
Q4. Explain use of group by clause in SQL with example.
Q5. What is the difference b/w Cartesian join and natural join.

Essay type questions:

- Q1. Explain SQL joins with taking suitable examples of tables.
Q2. What are aggregate functions? How we use aggregate functions? Give an example of each.
Q3. Explain the use of where, group by and having clause in a single SQL query? Give a suitable example.
Q4. Consider following schema given.
students(Roll_no, Sname, age, phone, address, class)
Classes(Class_name, CRoomNo, CStrength) and write an SQL syntax for.
1) Find student names of those 5th class students sitting in room number F-12.
2) Find number of students of 10th class living in Ajmer.
Q5. What do you mean by Sub queries? Why sub queries are useful.
Explain use of sub queries in set comparison.

Answers key for objective questions

- Q1: d Q2: c Q3: a Q4: b Q5: c