SQL Revision

SQL stands for Structured Query Language. It is the primary language to interact with databases. With the help of SQL, we can extract data from a database, modify this data and also update it whenever there is a requirement.

**Table creation command**

CREATE TABLE employee (  
 employee\_id INT PRIMARY KEY,   
 name VARCHAR (25),   
 age INT,  
 gender VARCHAR (25)  
);

**Deleting table**

**DROP TABLE employee**;

**Renaming table**

ALTER TABLE employee  
 RENAME TO emp\_table;

**Deleting row from a table**

DELETE FROM employee  
 WHERE *condition*;

**Create a new database**

CREATE DATABASE mydb;

# **Normalization in SQL**

Normalization is used to decompose a larger, complex table into simple and smaller ones. This helps us in removing all the redundant data. It is better to divide this complex table into multiple smaller tables which contains only unique information.

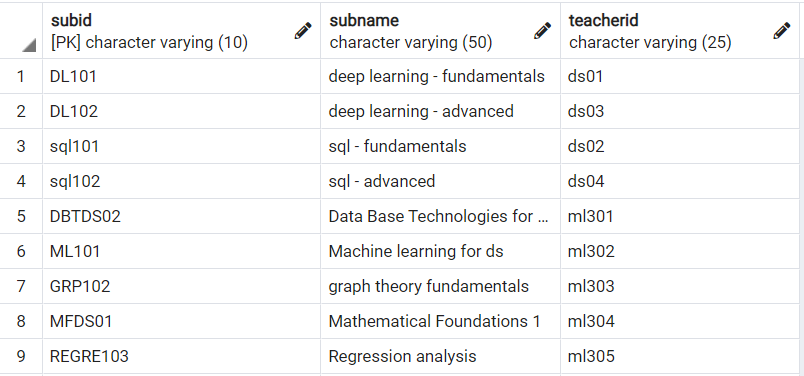
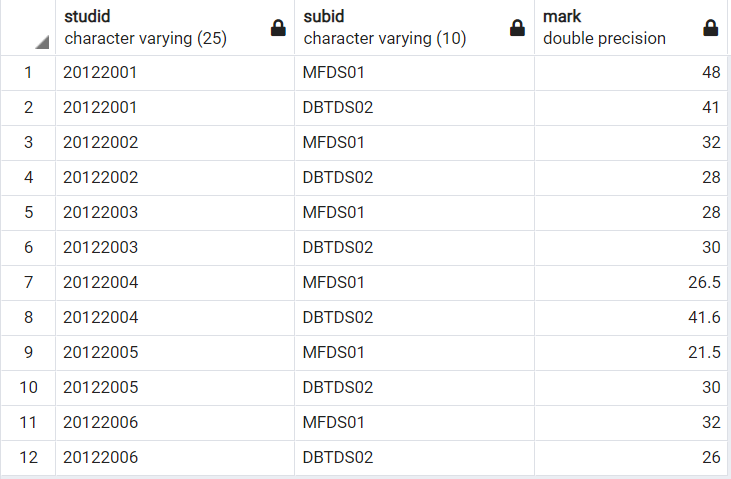
* First normal form
  + All attributes in the relation(table) are atomic
  + Unique column names
  + The order in which data is stored does not matter
  + Values should be of same type
* Second normal form
  + It is in first normal form
  + No partial dependency exists between non-key attributes and key attributes.
  + Note: suppose R (A B C D),   
    here A+B 🡪 D, B 🡪 C,   
    So, here AB is a candidate key, A, B are prime attributes and C, D are non-prime attributes. So, in second normal form there should not be any partial dependency. (Here C instead of depending on entire prime attributes, it is depending on a part of it (B)). If B becomes null, we cannot compute C.
* Third normal form
  + Should be in second normal form
  + A functional dependency from alpha to beta is called transitive if alpha, beta belongs to non-prime attributes.
  + R (A B C) where, A 🡪 B and B 🡪 C, here B and C are non -prime and there is a functional dependency between B 🡪 C.

# **SQL Joins**

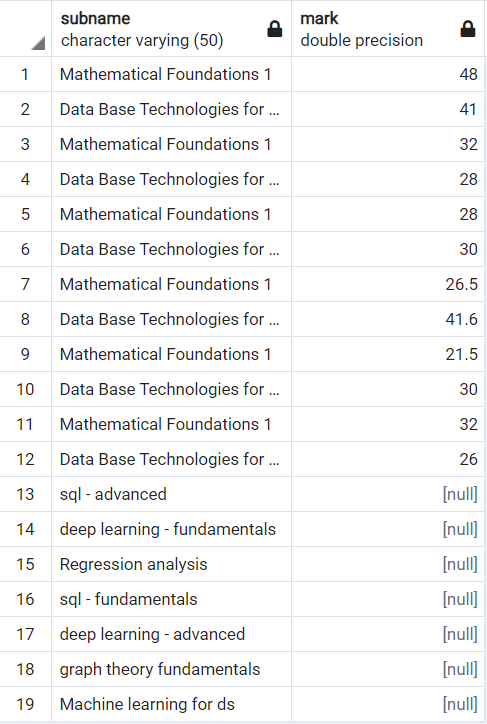
* INNER JOIN – Returns rows when there is a match in both tables
* LEFT JOIN – Returns all rows from the left table and the rows for which there is match on both tables
* RIGHT JOIN – Returns all rows from the right table, and rows which have matches on both tables.
* FULL OUTER JOIN – Returns rows which have matches on both tables and all the rows of left table and the right table as well.
* SELF JOIN – Used to join a table to itself as if the table were two tables.

**LEFT JOIN: select everything from tableA which are exclusive to tableA and the matches with tableB**.

SELECT \* FROM tableA  
 LEFT OUTER JOIN tableB  
 ON tableA.col\_match = tableB.col\_match

**Subject table Mark Table**

**Mark table**

SELECT subname, mark   
FROM subject  
LEFT JOIN mark  
ON mark.subid = subject.subid;

SELECT subname, mark  
FROM mark  
RIGHT JOIN subject  
ON mark.subid = subject.subid

**SQL Server** is a relational database management system. There are four types of DBMS;

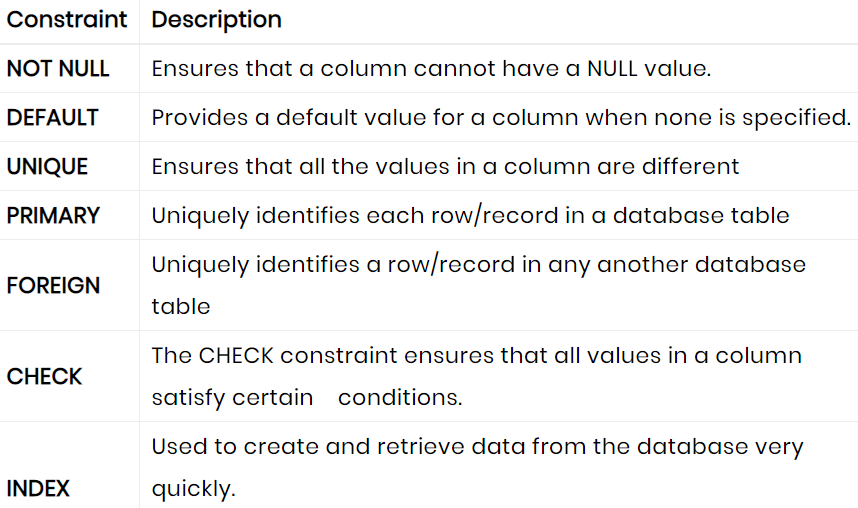
* Hierarchical
* Network
* Relational
* Object-Oriented

**Inserting data into table**

INSERT INTO table\_name (col\_1, col\_2, …) VALUES (value1, value2, …);

**Constraints in SQL**

Constraints are the rules enforced on data columns on a table. These are used to limit the type of data that can go into a table. Constraints can either be column level or table level.



Note: Primary key should be unique and not null.

*-- Viewing SQL tables*

Show tables

*--**Selecting tables from a database*

SELECT table\_name from information\_schema.tables  
 WHERE table\_schema = “public”;

***--****Selecting attributes from relation*

SELECT column\_name FROM information\_schema.columns  
 WHERE table\_name = “*table\_name*”;

--Selecting the user of the database

SELECT USER;

UPDATE COMMAND – DML(data manipulation language)

UPDATE employe\_tbl  
 SET last\_name = ‘Paul’  
 WHERE emp\_id = ‘EM101’;

**Renaming an attribute in a relation**

ALTER TABLE table\_name  
 RENAME COLUMN *col\_name* TO *new\_col\_name*

*Sql functions*

1. *Avg() –*
2. *Count() –*
3. *First() – returns first value of the selected column*
4. *Min()*
5. *Max()*
6. *Group by()*
7. *Having() :*

select store\_id, count (\*) from customer  
group by store\_id  
having count (\*)>13234;

1. *Upper() :*

select upper(first\_name) from worker;

1. *Lower():*

select lower(first\_name) from worker;

1. *Substring() –*

Select substring(first\_name,1,3) from worker;

1. ORDER BY – ASC , DESC

CASCADE – will propagate the change when the parent changes. (If you delete a row, rows in constrained tables that reference that row will also be deleted, etc.)

SET NULL – sets the column value to NULL when a parent row goes away.

RESTRICT – causes the attempted DELETE of a parent row to fail.

Subqueries – is a query within another SQL query, enclosed within parenthesis.

UPDATE Customers  
 SET Salary = Salary \* 0.25  
 WHERE Age IN (SELECT Age FROM Customers\_bkp WHERE  
 Age >=27);

Delete from customer  
 where age in (select age from customer\_bkp  
 where age >=28);

Insert into customer\_bkp  
 select \* from customers  
 where id in (select id  
 from customers);

EXCEPT – Used to combine two select statements and returns rows from the first select statement that are not returned by the second select statement. This means that EXCEPT returns only rows, which are not available in the second SELECT statement.

TEMPORARY TABLES – Stores a subset of data from a normal table for a certain period of time. Temporary tables are stored inside ‘tempdb’ which is a system database. They will be deleted when the current client session terminates. Temporary tables helps us to store and process intermediate results. There are two types of temporary tables : local and global : local temp table is available only for the session that has created it. Global temp tables are visible to all connections and dropped when the last connection referencing the table is closed.

* Used in stored procedure
* For storing results of a select into statement
* Storing the data within a session

CREATE TEMPORARY TABLE

VIEWS

It is like a virtual table that contains data from one or multiple tables. It does not hold any data and does not exist physically in the database. Similar to sql table, the view name be unique in a database.

* Helps to join and simplify multiple tables into single virtual table
* Can act as aggregated tables
* Views helps us to hide complexity of data
* Takes only little space to store, database contains only the definition of a view

Create view view\_name as  
 select statement

SQL Wildcard Characters

Used to substitute one or more characters in a string.

|  |  |  |
| --- | --- | --- |
| symbol | description | Example |
| \* | Represents zero or more characters | Bl\* finds bl, black, blue, blob etc. |
| ? | Represents a single character | h?t finds hot, hat and hit |
| [] | Represents any single character within the brackets | h[oa]t finds hot, and hat but not hit |
| ! | Represents any character not in the brackets | h[!oa]t finds hit, but not hot and hat |
| # | Represents any single numeric character | 2#5 finds 205, 284 etc. |
| % | Represents zero or more characters | bl% finds bl, black, blue etc |
| \_ | Represents a single character | h\_t finds hot, hat etc. |
| - | Represents a range of character | c[a-b] finds cat and cbt |
| ^ | Represents any character not in the brackets | h[^oa]t finds hit, but not hot and hat |

LIKE and ILIKE – Both are used for pattern matching, where LIKE is case sensitive and ILIKE is case insensitive.

SQL Aliases –

SQL Aliases are used to give a table, or a column in a table a temporary name. They are often used to make column names more readable. An alias only exists for the duration of that query. It is created with AS keyword.

IS NULL and IS NOT NULL are the keywords for checking null values

CASE statement goes through conditions and returns a value as per the conditions

SELECT Customer\_id,  
 CASE  
 WHEN Address\_id = 6 THEN 'NOT ALLOWED'  
 ELSE 'ALLOWED'  
 END AS Address  
FROM Customer;

COALESCE function returns the first non-null value in a list.

We can use CASE function for handling nulls as well;

SELECT  
 CASE WHEN expression IS NULL  
 THEN replacement  
 ELSE expression  
 END AS column alias;

EXISTS – logical operator that evaluates a subquery for existence of records. If the subquery has one or more records, EXISTS returns TRUE, else it returns FALSE. Since EXISTS returns a logical value, it is common to use EXISTS in the WHERE clause, which expects logical expressions.

SELECT stname  
 FROM student  
 WHERE EXISTS (  
 SELECT \* FROM mark  
 WHERE mark. studid = student. studid);

ALL and ANY operators

SELECT \* FROM student  
 WHERE age > ALL (SELECT age FROM lecturer);

SELECT \* FROM student  
 WHERE age > ANY (SELECT age FROM lecturer)

SELECT INTO statement copies data from one table into a new table.

SELECT \* INTO newtable IN mydb  
 FROM customer;

INSERT INTO SELECT statement copies data from one table and inserts it into another table. INSERT INTO SELECT statement requires that the data types in source and target tables matches.

Insert into table2  
 select \* from table1  
 where condition;

Difference between SELECT INTO and INSERT INTO SELECT:  
 First for the insert you need to pre-define the destination table. SELECT INTO created the table as part of the statement. Type conversion errors can happen with the insert statement.

Stored Procedure:

A stored procedure is a prepared SQL code that you can save, so the code can be reused over an over again. We can also pass parameters to a stored procedure, so that the stored procedure can act based on the parameter value(s) that is passed.

SQL Hosting

If we want out web site to be able to store and retrieve data from a database, our web server should have access to a database-system that uses the SQL language.

Materialized views

The basic difference between View and Materialized View is that Views are not stored physically on the disk. ... View can be defined as a virtual table created as a result of the query expression. However, Materialized View is a physical copy, picture or snapshot of the base table.

A normal view is a query that defines a virtual table -- you don't actually have the data sitting in the table, you create it on the fly by executing.

A materialized view is a view where the query gets run and the data gets saved in an actual table.

The data in the materialized view gets refreshed when you tell it to.

A couple use cases:

* We have multiple Oracle instances where we want to have the master data on one instance, and a reasonably current copy of the data on the other instances. We don't want to assume that the database links between them will always be up and operating. So we set up materialized views on the other instances, with queries like select a,b,c from mytable@master and tell them to refresh daily.
* Materialized views are also useful in query rewrite. Let's say you have a fact table in a data warehouse with every book ever borrowed from a library, with dates and borrowers. And that staff regularly want to know how many times a book has been borrowed. Then build a materialized view as select book\_id, book\_name, count(\*) as borrowings from book\_trans group by book\_id, book\_name, set it for whatever update frequency you want -- usually the update frequency for the warehouse itself. Now if somebody runs a query like that for a particular book against the book\_trans table, the query rewrite capability in Oracle will be smart enough to look at the materialized view rather than walking through the millions of rows in book\_trans.

Usually, you're building materialized views for performance and stability reasons -- flaky networks, or doing long queries off hours.

Index

Partition

BTREE – data structure

Used in mongo db – index