SQL

What is SQL?

SQL is short for Structured Query Language and is used for storing, manipulating and retrieving data. It was invented by IBM. Many of the big league companies have large amounts of data and SQL helps to control access to this data and manipulate data to get business insights from the data.

Who uses SQL?

SQL is used by many types of people; a few important ones are: The Software developers who create applications or programs which have databases at the backend. The Database managers or the administrators who manage the data of these huge companies. The Business managers who use the data to find out key insights to make business decisions.

What is a Database?

Database is a tool for storing and retrieving data effectively and efficiently. There are 2 parts to the data being stored: The data itself which will be stored in the form of tables. The other is the metadata which defines and describes the data structure.

Brief about Tables

A record in a table is a row in the table. A column in the table is an attribute in the table which contain the same type of values for all the records. The intersection of these rows and columns is called a cell and it contains a unique single value.

What is a DBMS?

DBMS is short for Database Management Systems and is used for defining, managing and process databases. It does the following:

- Creation of a new database and their data structures
- Allows modification of the data
- Allows retrieval of the data
- Allows storage of the data over a long period of time
- Enables recovery in times of failures
- Controls access of the users

Types of SQL Commands

SQL commands are classified into 5 different commands. These commands are:

- DDL – Data Definition Language
    - Enables to define the structure of the data
    - Can create, change or delete the existing elements of the database
    - Elements includes tables, views, schemas, etc.
    - A few commands:
        - CREATE – to create a table or a database
        - ALTER – to alter any of the columns of the table
        - DROP – to delete a particular table from the database
    - Used by Database designers and architects
- DML – Data Manipulation Language
    - Enables to manipulate the data contained in the database
    - Can add, change or delete data from the table
    - A few commands:

        - INSERT – to insert a record in the table
        - UPDATE – to change values already inserted in the record
        - DELETE – to delete particular rows of the table
    - Used mainly by Operations and Database managers
- DQL – Data Query Language
    - Used to retrieve stored data
    - A few commands
        - SELECT – selects the table from which the data is to be fetched
        - ORDER BY – orders the table in ascending or descending order
        - GROUP BY – groups the records according to the condition
    - Used mainly by the business users or the analysts
- DCL – Data Control Language
    - Used to control the access privileges of the databases of the users
    - A few commands:
        - GRANT – this will grant privileges to a particular user
        - REVOKE – this will take the privileges away from the user
    - Mainly used by the Database administrators or the owners of the database
- TCC – Transaction Control Commands
    - Used to manipulate the different transactions over the database
    - Used to maintain the data integrity
    - A few commands:
        - COMMIT – makes the changes done to the database permanent
        - ROLLBACK – rolls back the database to the previous save point
    - Used by the Operations and Database managers to maintain the integrity of the database

About PostgreSQL

PostgreSQL is an advanced Object-Relational database management system. A few companies which use PostgreSQL are: Instagram, Spotify, Netflix, Uber, Instacart, Reddit, etc. PostgreSQL is open source, has complete ACID compliance, has comprehensive documentation and active discussion forums. It is best used by Data Analysts as they tend to work with complex queries and it is easier to execute in PostgreSQL. PostgreSQL is best suited for data warehousing and data analysis applications. PostgreSQL is supported by all major cloud service providers such as: Amazon, Google, Microsoft, etc.

Basic SQL Commands

- CREATE
  Syntax: create table table_name (column_1 data-type constraints, column_2 data-type constraints, … column_n data-type constraints, table_constraints);

- INSERT
  Syntax: insert into table_name (column_1, column_2, …) values (value_1, value_2, …);

  Or

  Insert into table_name values (value_1, value_2, …);

- COPY
  Syntax: copy table_name (column_1, column_2, …) from ‘address of file’ delimiter ‘*delimiter’* type_of_file header;

- SELECT
  Syntax: select column_1, column_2, … from table_name;
  Or
  select * from table_name;

- SELECT DISTINCT
  Syntax: select distinct column_1, column_2, … from table_name;

- WHERE
  Syntax: select column_name from table_name where condition;

- UPDATE
  Syntax: update table_name set column_1 = value_1, column_2 = value_2, … where condition;

- DELETE
  Syntax: delete from table_name where condition;

- ALTER
  Syntax: alter table table_name *specify actions;*
  Actions: Add, delete (drop), Modify or rename columns
  Add, drop constraints
  Add, drop index

- IN
  Syntax: select * from table_name where column_name in (value_1, value_2, …);

- BETWEEN
  Syntax: select column_name from table_name where column_name between value_1 and value_2;

- LIKE
  Syntax: select column_name from table_name where column_name like *wildcards;*
  Wildcards: %, _

- ORDER BY
  Syntax: select column_name from table_name where *conditions* order by column_name asc/desc;

- LIMIT
  Syntax: select column_name from table_name where *conditions* order by *expressions* asc/desc limit row_count;

- AS
  Syntax: select column_name as column_alias from table_name;

- GROUP BY
  Syntax: select column_name from table_name group by column_name;

- HAVING
  Syntax: select column_name from table_name where *conditions* group by column_name having *conditions*;

- CASE WHEN
  Syntax: case when *conditions* then result when … else result end;
  case expression when value then result, when …, else result end;

Aggregate functions

- COUNT
  Syntax: select count(column_name) from table_name;

- SUM
  Syntax: select sum(column_name) from table_name;

- AVERAGE
  Syntax: select avg(column_name) from table_name;

- MIN
  Syntax: select min(column_name) from table_name;

- MAX
  Syntax: select max(column_name) from table_name;

Constraints

- NOT NULL – Ensures the column does not have null values
- DEFAULT – Provides a default value for a column when no value is given
- UNIQUE – Ensures that all values in a column are different
- CHECK – makes sure that all values in the column satisfy a certain condition
- PRIMARY KEY – Used to uniquely identify a row in a table
- FOREIGN KEY – Used to ensure referential integrity of the data

Primary Key

A primary key is used to uniquely identify a row in a table. It ensures that there are no duplicate records in a table. It can consist of one or more than one columns. No value in this field can be NULL or blank.

Foreign Key

A foreign key is used to maintain referential integrity of the table. It is usually a column which references another column of another table.

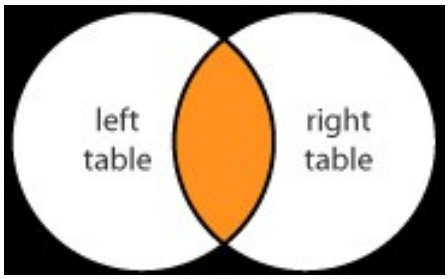
To comment in SQL

--	Single line comment
/* */	Multiple line comment

Joins

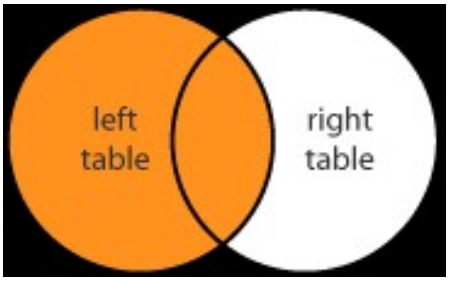
We join two tables on the basis of a common column. To join 2 tables, we must know the names of the tables to be joined, the common column based on which we will join the tables and the list of columns from each table. There are different types of joins, such as:

- Inner Join
  Result contains only those records which have the same primary key value.



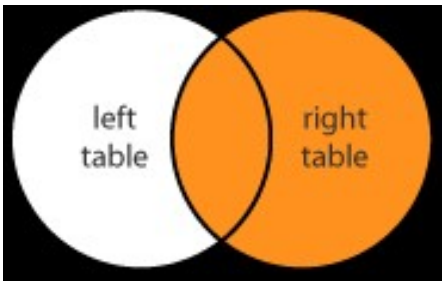
  Syntax: select columns from table_1 inner join table_2 on table_1.column_name = table_2.column_name;

- Left Join
  Result contains all the records of the primary key value of the left table.
  All values of left table are shown.
  Values of right table against left table are shown, if not, then NULL values are shown.



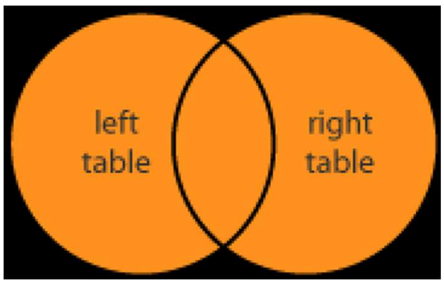
  Syntax: select table_1.column_1, table_2.column_2, … from table_1 left join table_2 on table_1.common_field = table_2.common_field;

- Right Join
  Results contains all the records of the primary key value of the right table.
  All values of right table are shown.
  Values of left table against right table are shown, if not, then NULL values are shown



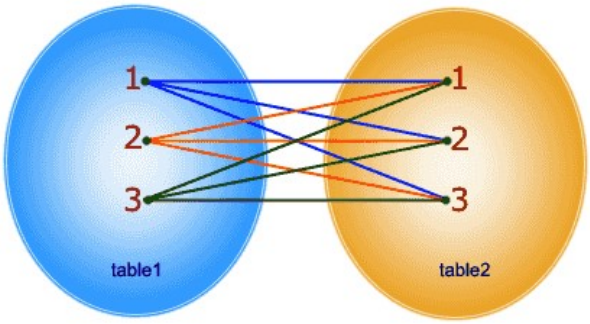
Syntax: select table_1.column_1, table_2.column_2, … from table_1 right join table_2 on table_1.common_field = table_2.common_field;

- Full Outer Join
  Results contains the values of both the left and right outer joins.



Syntax: select table_1.column_1, table_2.column_2, … from table_1 full join table_2 on table_1.common_field = table_2.common_field;

- Cross Join
  It creates a Cartesian product between the two sets of data.



Syntax: select table_1.column_1, table_2.columns_2, … from table_1, table_2, …;

Combining Queries

Combining similar data using operators like Union, Intersect and Exceptions are called combining queries.

- INTERSECT
  The intersect command will give the intersection of the two or more tables as defined, as the result.

Syntax: select column_1, column_2, … from table_1 intersect select column_1, column_2, … from table_2;

- EXCEPT

  The except command will return all the records from the first table which are not returned by the second table.

  Syntax: select expression_1, expression_2, … from tables where *conditions* except select expression_1, expression_2, … from tables where *conditions;*

- UNION

  The union command will combine the results of two or more select statements while removing the duplicate rows between the various select statements.

  Syntax: select expression_1, expression_2, … from tables where *conditions* union select expression_1, expression_2, … from tables where *conditions;*

Sub-Queries

Subquery is a query within a query. These subqueries can be given in the Where clause, From clause or the Select clause.

- Sub-Query in the WHERE clause
  Syntax: select column_1 from table_1 where column_2 *comparison-operator* (select column_3 from table_2 where *conditions*);

- Sub-Query in the FROM clause
- Sub-Query in the SELECT clause

Rules for using Sub-Queries

i. Subqueries must be enclosed within a parenthesis.
ii. A subquery can have only one column in the select clause, unless main query has multiple columns for it to compare.
iii. An order by command can't be used in a subquery although the main query can use it. The group by command can be used for the same purpose as the order by command in a sub-query.
iv. Subqueries that return more than one record can be used with multiple values operators such as IN.
v. The select list can't include references to values that evaluate to a BLOB, ARRAY, CLOB or NCLOB.
vi. A subquery can't be immediately enclosed in a set function.
vii. The between operator can't be used with a subquery BUT the between operator can be used within a subquery.

Views

View is not a physical table but a virtual table created by a query joining one or more tables. It provides certain benefits such as ease of use, saves space and provides better security.

Syntax: create/replace view view_name as select column_name from table_name where *conditions;*

We can drop (delete) a view and even update a view with new definitions using the DROP and UPDATE commands respectively. View can only be updated under certain conditions which are:

i. The select clause may not contain the keyword DISTINCT.
ii. The select clause may not contain aggregate functions.
iii. The select clause may not contain set functions.
iv. The select clause may not contain set operators.
v. The select clause may not contain an ORDER BY clause.
vi. The from clause may not contain multiple tables.
vii. The where clause may not contain subqueries.
viii. The query may not contain group by and having clauses.
ix. Calculated columns may not be updated.
x. All NOT NULL columns from the base table must be included in the view in order for the INSERT query to function.

Index

An index is a performance tuning method that allows faster retrieval of records from the database. An index creates an entry for each value that appears in the indexed columns. A simple index is an index on a single column, while a composite index is an index on two or more columns. We can even drop or alter index.

Syntax: create *unique* index index_name on table_name (index_column_1 asc/desc, index_column_2 asc/desc …);

A few good practices when indexing:

- Build index on columns of integer types.
- Keep index as narrow as possible.
- Column order is important.
- Make sure the column you are building an index for is declared NOT NULL.
- Build an index only when necessary.

SQL Functions

SQL Functions are classified on the basis of the data types they are applied on. The classifications are:

- String Functions (Textual data)
  - Length
    - Returns the length of the specified string expressed as number of characters.
    - Syntax: length(string)
  - Upper & Lower
    - Converts the specified characters to uppercase or lower case.
    - Syntax: upper(string) or lower(string)
  - Replace

        - Replaces all occurrences of a set of characters to some other set of characters.
          It is case sensitive.
        - Syntax: replace(string, from_substring, to_substring)
    - Trim
        - It removes all specified characters either from the beginning or the end of a string.
        - Syntax: trim([leading | trailing | both] [trim_character] from string)
    - LTrim
        - It removes all specified characters from the Left hand side of the string.
        - Syntax: ltrim (string, trim_character)
    - RTrim
        - It removes all specified characters from the Right hand side of the string.
        - Syntax: rtrim (string, trim_character)
    - Substring
        - Used to extract a substring from a string
        - Syntax: substring (string [from start_position] [for length])
    - Concatenation
        - Allows you to join 2 or more strings together.
        - Syntax: string_1 || string_2 || … string_n
    - String aggregation
        - It concatenates input values into a string separated by a delimiter.
        - Syntax: string_agg (expression, delimiter);
- Mathematical Functions (Numerical Data)
    - Setseed
        - It will return a repeatable sequence of random numbers that is derived from the seed.
        - Seed can have a value between 1.0 to -1.0 inclusive.
        - Syntax: setseed(seed)
    - Round
        - Returns a number rounded to a certain number of decimal places.
        - Syntax: round(number)
    - Power
        - Returns m raised to the nth power.
        - Power (m, n)
    - Ceil
        - Returns the smallest integer value that is greater than or equal to a number.
        - Syntax: ciel(number)
    - Floor
        - Returns the largest integer value that is equal or less than a number.
        - Syntax: floor(number)
    - Random
        - It generates a number between 0 and 1 wherein 0 is included but 1 is excluded.
        - Syntax: select random()*(b-a)+a; [When a is included and b excluded]
        - Syntax: select floor (random()*(b*a-1))+a; [When both a and b is included]
- Date-Time Functions (Date and Time data)
    - Current Date

        - Returns the current date
        - Syntax: current_date
    - Current Time
        - Returns the current time within the time zone
        - Syntax: current_time ([precision])
    - Current Timestamp
        - Returns the current date and time within the time zone
        - Syntax: current_timestamp ([precision])
    - Age
        - Returns the number of years, months and days between two dates
        - Syntax: age (date_1, date_2)
    - Extract
        - It extracts parts from a date.
        - Syntax: extract(unit from date);
        - Units are:

Unit	Explanation
day	Day of the month (1 to 31)
decade	Year divided by 10
doy	Day of the year (1=first day of year, 365/366=last day of the year, depending if it is a leap year)
epoch	Number of seconds since '1970-01-01 00:00:00 UTC', if date value. Number of seconds in an interval, if interval value
hour	Hour (0 to 23)
minute	Minute (0 to 59)
month	Number for the month (1 to 12), if date value. Number of months (0 to 11), if interval value
second	Seconds (and fractional seconds)
year	Year as 4-digits

- Pattern Matching
    - Like statements
        - Same as before
        - It matches on whole string
    - Similar to statements
    - Regular Expressions (REGEX)
        - Wild cards for REGEX is different than LIKE statements
        - Regex can match on part of strings also
        - Wild cards of regex are:

Wildcard	Explanation
|	Denotes alternation (either of two alternatives).
*	Denotes repetition of the previous item zero or more times
+	Denotes repetition of the previous item one or more times.
?	Denotes repetition of the previous item zero or one time.
{m}	denotes repetition of the previous item exactly m times.
{m,}	denotes repetition of the previous item m or more times.
{m,n}	denotes repetition of the previous item at least m and not more than n times
^,$	^ denotes start of the string, $ denotes start of the string
[chars]	a *bracket expression*, matching any one of the chars
~*	~ means case sensitive and ~* means case insensitive

- Data type conversion
  - Numeric/Date time to String
    - Converts a number or date to a string
    - Syntax: to_char(value, format_mask);

Format Mask

Parameter	Explanation
9	Value (with no leading zeros)
0	Value (with leading zeros)
.	Decimal
,	Group separator
PR	Negative value in angle brackets
S	Sign
L	Currency symbol
MI	Minus sign (for negative numbers)
PL	Plus sign (for positive numbers)
SG	Plus/minus sign (for positive and negative numbers)
EEEE	Scientific notation

Format Mask

Parameter	Explanation
YYYY	4-digit year
MM	Month (01-12; JAN = 01).
Mon	Abbreviated name of month capitalized
Month	Name of month capitalized, padded with blanks to length of 9 characters
DAY	Name of day in all uppercase, padded with blanks to length of 9 characters
Day	Name of day capitalized, padded with blanks to length of 9 characters
DDD	Day of year (1-366)
DD	Day of month (01-31)
HH	Hour of day (01-12)
HH12	Hour of day (01-12)
HH24	Hour of day (00-23)
MI	Minute (00-59)
SS	Second (00-59)
am, AM, pm, or PM	Meridian indicator

  - String to Numeric/Date time
    - Converts a string to date
    - Syntax: to_date(string_1, format_mask);
    - Syntax: to_number(string_1, format_mask);
- User Access Control
  - Create user
    - Creates a database account that allows you to log into the database
    - Syntax: create user user_name [with password ‘password_value’ | valid until ‘expiration’];
  - Grant access
    - Syntax: grant privileges on object to user;

- Revoke access
  - Syntax: revoke privileges on object from user;

Privileges

Privilege	Description
SELECT	Ability to perform SELECT statements on the table.
INSERT	Ability to perform INSERT statements on the table.
UPDATE	Ability to perform UPDATE statements on the table.
DELETE	Ability to perform DELETE statements on the table.
TRUNCATE	Ability to perform TRUNCATE statements on the table.
REFERENCES	Ability to create foreign keys (requires privileges on both parent and child tables).
TRIGGER	Ability to create triggers on the table.
CREATE	Ability to perform CREATE TABLE statements.
ALL	Grants all permissions.

- Drop user
  - Removes a user from the database
  - Syntax: drop user user_name;
- Rename user
  - Used to rename a user in database
  - Syntax: alter user user_name rename to new_name;
- Find logged in users
  - Syntax: select distinct user_name from pg_stat_activity;

Table Spaces

Table spaces allow database administrators to define locations in the file system where the files representing the database objects can be stored. It can be used by only a database superuser.

Syntax: create tablespace tablespace_name location location_on_drive;

ACID Compliance

ACID is an acronym for Atomicity, Consistency, Isolation and Durability. Atomicity states that the unit of transaction is atomic in nature and either happens completely or does not happen. Consistency ensures that a transaction can bring a database from one valid point to another. Isolation keeps transactions separated until they are finished. Durability guarantees that the database will keep records of all updates which can be called for anytime.

Truncate

It removes all records from the table or a set of tables. It performs the same function as a delete statement but without a where clause.

Syntax: truncate only table_name cascade/restrict;

Normalization

It is a technique of organising the data in the database. It is used to reduce redundancy in the database. Redundancies lead to 3 types of anomalies:

- Insert anomaly
- Update anomaly
- Delete anomaly

There are multiple types of Normal forms, which are:

- First Normal Form
  - Each attribute should contain a single value in each row.
  - Maintain data types in a column.
  - Each column should have a unique column name.
- Second Normal Form
  - Table should be in First Normal Form.
  - There should not be any partial dependency.
- Third Normal Form
  - Table should be in Second Normal Form.
  - There should be no transitive dependency.
- Boyce-Codd Normal Form
  - Table should be in the Third Normal Form.
  - If B depends on A, then A should be a superkey.
- Fourth Normal Form
  - Table should be in Boyce-Codd Normal Form.
  - There should not be any multivalued dependency.
- Fifth Normal Form
  - Table should be in the Fourth Normal Form.
  - There should not be any join dependency.

Schema

A schema is a collection of database objects associated with one particular database. One database can have multiple schemas. Schema is used for allowing many users to use one database without interfering with each other. It is used to make database more organized and manageable. Third party applications can be put into separate schemas so they do not collide with the names of other objects. Syntax for creating a schema: create schema schema_name;

Best practices in SQL

1. Use the Explain command. The explain command will display the execution plan for a query statement without running the query. Syntax: explain *query*;
2. There is a difference between Soft Delete and Hard Delete. Soft Delete marks the record as deleted without actually deleting the record from the database whereas Hard delete actually deletes the record from the database.
3. Using Update statements will lead to excess data storage size usage/ One can use other statements such as the case statement.
4. Vacuum command reclaims the space occupied by the soft deleted records. Syntax: vacuum table_name;
5. Using a truncate statement is more efficient than the delete statement.
6. Using LIKE statements in place of REGEX statements is preferred.
7. Rather than using SIMILAR TO statements, using LIKE and REGEX is advised.
8. Avoid unnecessary string operations such as upper, lower, replace, etc.
9. Use trim instead of replace and avoid unnecessary string columns.

10. Use subqueries to select only the required fields of the table.
11. Avoid one to many joins by using GROUP BY clause.