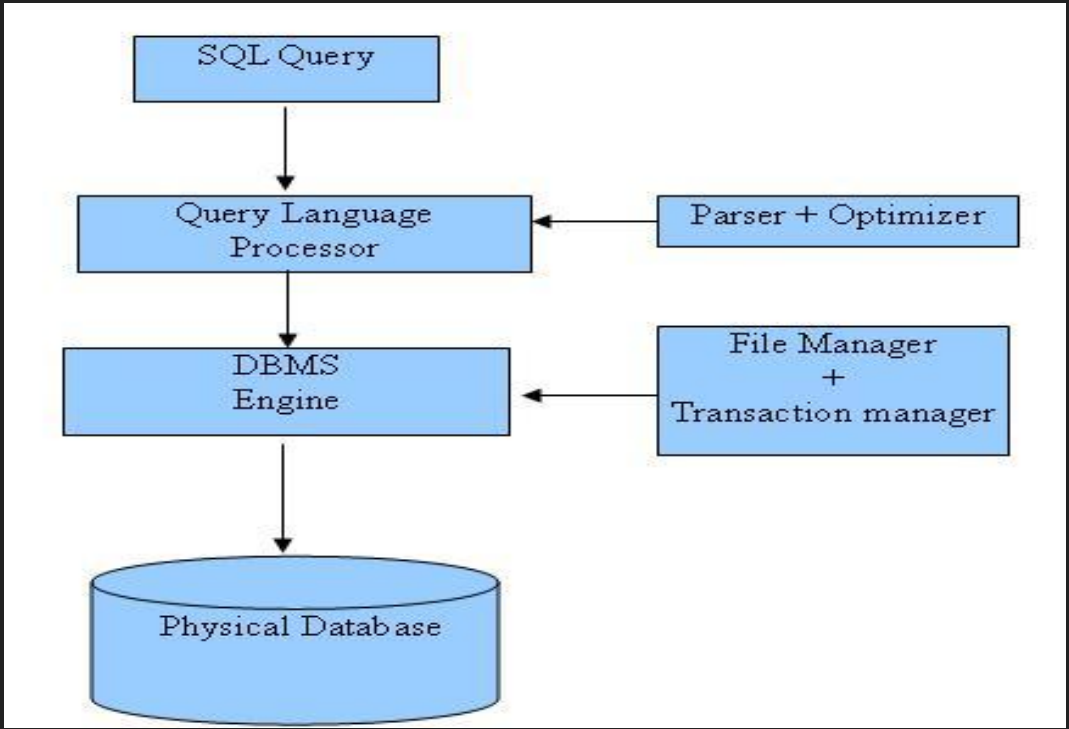
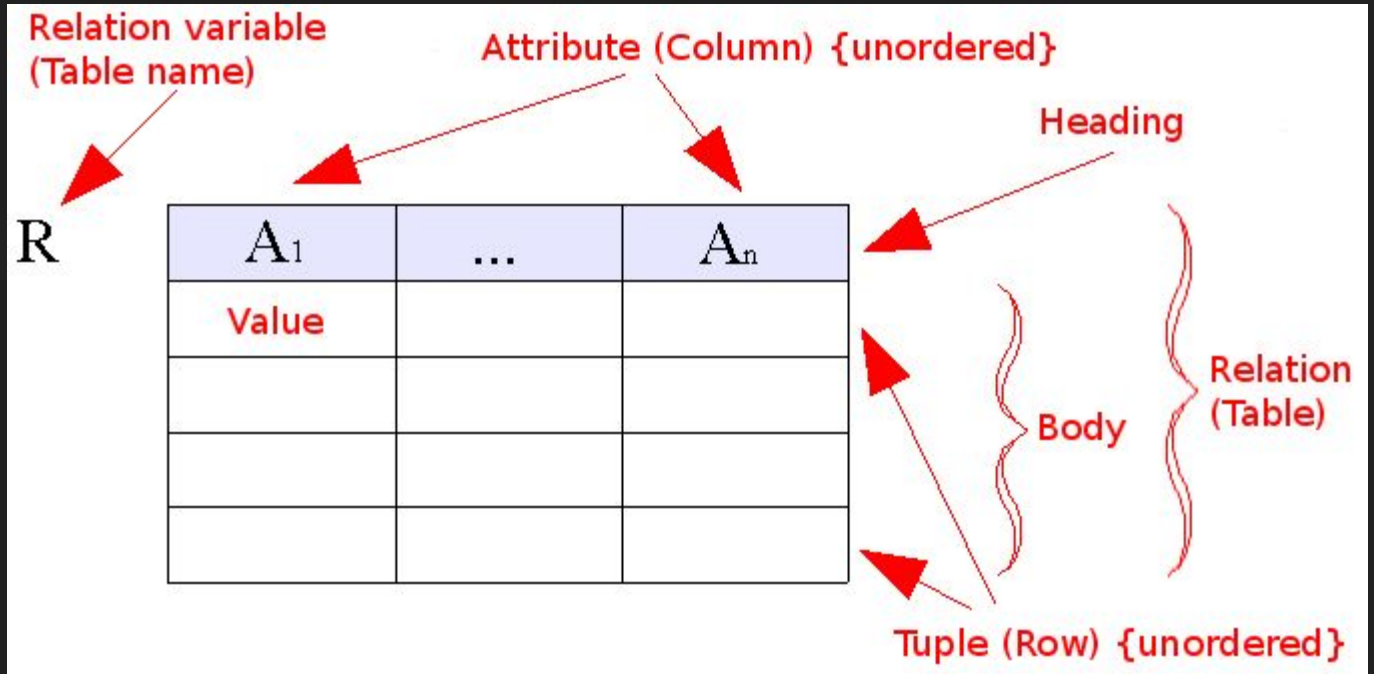
SQL is a database computer language used to store manipulate and retrieve the data from a relational database
Sql Is the standard language for relational database management system

DATABASE ARCHITECTURE



RDBMS:- is a database management system that is based on RELATIONAL MODEL
Table:- th data in the rdbms is stored in database objects called tables
Field or attribute:- every table is broken into smaller entities called fields
record
Record or row:- individual entry that exist in a table
Column:- vertical entity in a table that contain all information associate with specific field in a table
model



RDBMS stands for Relational Database Management System. RDBMS is the basis for SQL, and for all modern database systems like MS SQL Server, IBM DB2, Oracle, MySQL, and Microsoft Access.

What is a table?

The data in an RDBMS is stored in database objects which are called as tables.

What is a field?

Every table is broken up into smaller entities called fields. The fields in the CUSTOMERS table consist of ID, NAME, AGE, ADDRESS and SALARY.

What is a Record or a Row?

A record is also called as a row of data is each individual entry that exists in a table.

What is a column?

A column is a vertical entity in a table that contains all information associated with a specific field in a table.

SQL CONSTRAINTS

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. This ensures the accuracy and reliability of the data in the database.

Constraints can either be column level or table level. Column level constraints are applied only to one column whereas, table level constraints are applied to the entire table.

Following are some of the most commonly used constraints available in SQL −

- NOT NULL Constraint − Ensures that a column cannot have a NULL value.
- DEFAULT Constraint − Provides a default value for a column when none is specified.
- UNIQUE Constraint − Ensures that all the values in a column are different.
- PRIMARY Key − Uniquely identifies each row/record in a database table.
- FOREIGN Key − Uniquely identifies a row/record in any another database table.
- CHECK Constraint − The CHECK constraint ensures that all values in a column satisfy certain conditions.
- INDEX − Used to create and retrieve data from the database very quickly.

Data Integrity

The following categories of data integrity exist with each RDBMS −

- Entity Integrity − There are no duplicate rows in a table.
- Domain Integrity − Enforces valid entries for a given column by restricting the type, the format, or the range of values.
- Referential integrity − Rows cannot be deleted, which are used by other records.
- User-Defined Integrity − Enforces some specific business rules that do not fall into entity, domain or referential integrity.

The most important point to be noted here is that SQL is case insensitive, which means SELECT and select have same meaning in SQL statements. Whereas, MySQL makes difference in table names. So, if you are working with MySQL, then you need to give table names as they exist in the database.

Database Normalization

Database normalization is the process of efficiently organizing data in a database. There are two reasons of this normalization process −

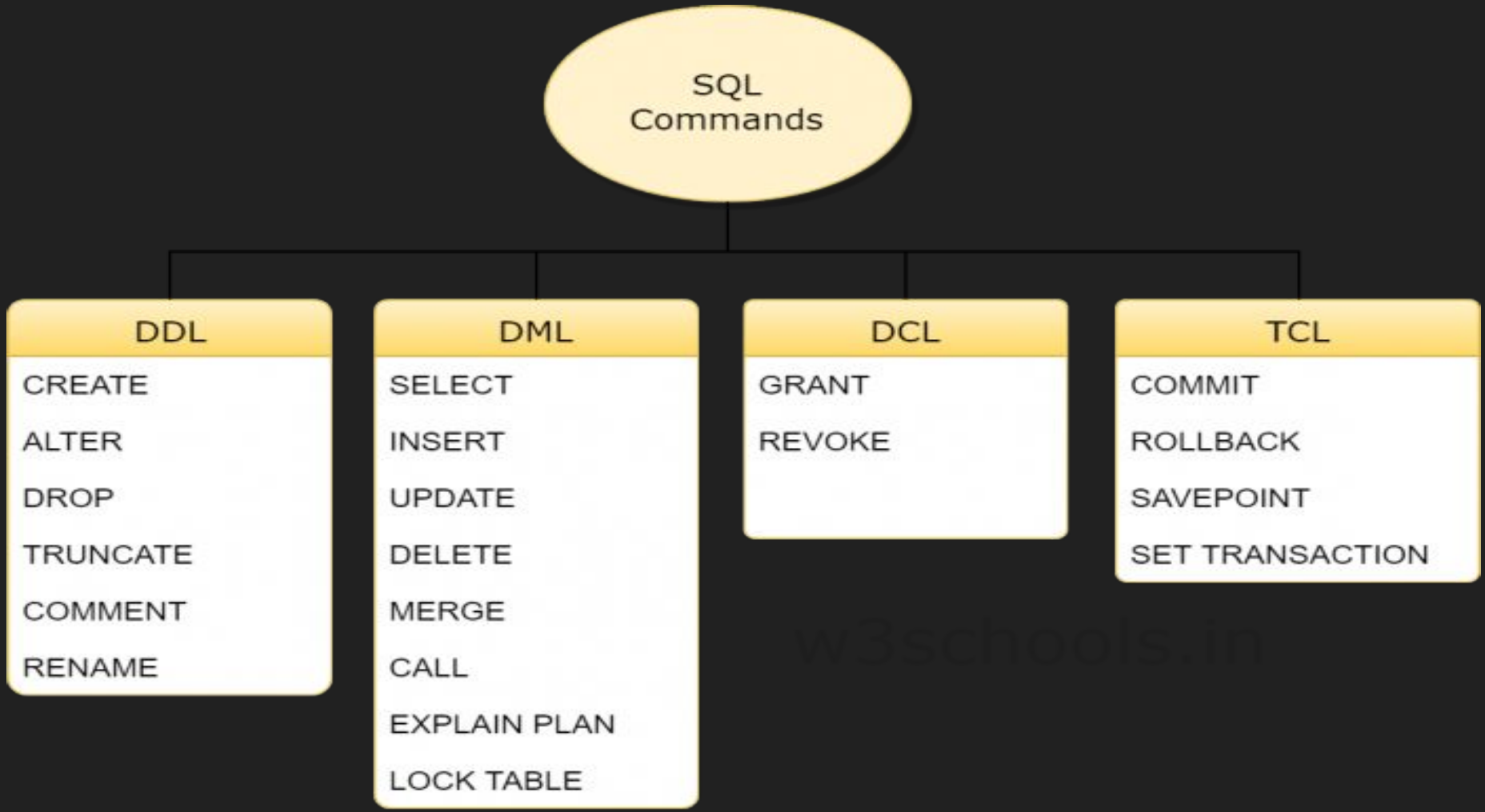
- Eliminating redundant data, for example, storing the same data in more than one table.
- Ensuring data dependencies make sense.

Both these reasons are worthy goals as they reduce the amount of space a database consumes and ensures that data is logically stored. Normalization consists of a series of guidelines that help guide you in creating a good database structure.

Normalization guidelines are divided into NORMAL FORMS; think of a form as the format or the way a database structure is laid out. The aim of normal forms is to organize the database structure, so that it complies with the rules of first normal form, then second normal form and finally the third normal form.

It is your choice to take it further and go to the fourth normal form, fifth normal form and so on, but in general, the third normal form is more than enough.

- First Normal Form (1NF)
- Second Normal Form (2NF)
- Third Normal Form (3NF)


SQL Commands
DDL
CREATE
ALTER
DROP
TRUNCATE
COMMENT
RENAME
DML
SELECT
INSERT
UPDATE
DELETE
MERGE
CALL
EXPLAIN PLAN
LOCK TABLE
DCL
GRANT
REVOKE
TCL
COMMIT
ROLLBACK
SAVEPOINT
SET TRANSACTION
w3schools.in

- Data Definition Language
- Data Manipulation Language
- Data Control Language
- Transaction Control Language

DDL is short name of Data Definition Language, which deals with database schemas and descriptions, of how the data should reside in the database.

CREATE - to create a database and its objects like (table, index, views, store procedure, function, and triggers)
ALTER - alters the structure of the existing database
DROP - delete objects from the database
TRUNCATE - remove all records from a table, including all spaces allocated for the records are removed
COMMENT - add comments to the data dictionary
RENAME - rename an object

DML is short name of Data Manipulation Language which deals with data manipulation and includes most common SQL statements such SELECT, INSERT, UPDATE, DELETE, etc., and it is used to store, modify, retrieve, delete and update data in a database.

SELECT - retrieve data from a database
INSERT - insert data into a table
UPDATE - updates existing data within a table
DELETE - Delete all records from a database table
MERGE - UPSERT operation (insert or update)
CALL - call a PL/SQL or Java subprogram
EXPLAIN PLAN - interpretation of the data access path
LOCK TABLE - concurrency Control

DCL

DCL is short name of Data Control Language which includes commands such as GRANT and mostly concerned with rights, permissions and other controls of the database system.

GRANT - allow users access privileges to the database
REVOKE - withdraw users access privileges given by using the GRANT command

TCL

TCL is short name of Transaction Control Language which deals with a transaction within a database.

COMMIT - commits a Transaction
ROLLBACK - rollback a transaction in case of any error occurs
SAVEPOINT - to rollback the transaction making points within groups
SET TRANSACTION - specify characteristics of the transaction

- SQL - RDBMS Concepts
- SQL - Databases
- SQL - Syntax
- SQL - Data Types
- SQL - Operators
- SQL - Expressions
- SQL - Create Database
- SQL - Drop Database
- SQL - Select Database
- SQL - Create Table
- SQL - Drop Table
- SQL - Insert Query
- SQL - Select Query
- SQL - Where Clause
- SQL - AND & OR Clauses
- SQL - Update Query
- SQL - Delete Query
- SQL - Like Clause
- SQL - Top Clause
- SQL - Order By
- SQL - Group By
- SQL - Distinct Keyword
- SQL - Sorting Results

There are different types of SQL expressions, which are mentioned below −

- Boolean Expressions
- Numeric Expressions(count ,sum)
- Date Expressions(date,smalldate)

```
SELECT * FROM CUSTOMERS
  ORDER BY (CASE ADDRESS
  WHEN 'DELHI'      THEN 1
  WHEN 'BHOPAL'   THEN 2
  WHEN 'KOTA'       THEN 3
  WHEN 'AHMEDABAD' THEN 4
  WHEN 'MP'    THEN 5
ELSE 100 END) ASC, ADDRESS DESC;
```

DATABASE

- SQL CREATE DATABASE Statement

CREATE DATABASE database_name;

- SQL DROP DATABASE Statement

DROP DATABASE database_name;

- SQL USE Statement

USE database_name;

- SQL TRANSACTION Statement

  Begin transaction;

- SQL COMMIT Statement

COMMIT;

- SQL ROLLBACK Statement

ROLLBACK;

TABLE

- SQL CREATE TABLE Statement

CREATE TABLE table_name(column1 datatype PRIMARY KEY(one or more columns) default value);

- SQL DROP TABLE Statement

DROP TABLE table_name;

- SQL CREATE INDEX Statement

CREATE UNIQUE INDEX index_name ON table_name (column1, column2,...columnN);

- SQL DROP INDEX Statement

ALTER TABLE table_name DROP INDEX index_name;

- SQL SELECT Statement

SELECT column1 FROM table_name;

- SQL DISTINCT Clause

SELECT DISTINCT column1 FROM table_name;

TABLE

- SQL WHERE Clause

SELECT column1 FROM table_name WHERE CONDITION;

- SQL AND/OR Clause

SELECT column1 FROM table_name WHERE CONDITION-1 {AND|OR}CONDITION-2;

- SQL IN Clause

SELECT column1 FROM table_name WHERE column_name NOT IN (val-1, val-2,...val-N);

- SQL HAVING Clause

SELECT SUM(column_name) FROM table_name WHERE CONDITION GROUP BY column_name HAVING (arithmetic function condition);

TABLE

- SQL BETWEEN Clause

SELECT column1 FROM table_name WHERE column_name BETWEEN val-1 AND val-2;

- SQL LIKE Clause

SELECT column1, FROM table_name WHERE column_name LIKE {_ap% PATTERN% };

- SQL ORDER BY Clause

SELECT column1,FROM table_name WHERE CONDITION ORDER BY column_name {ASC|DESC};

- SQL GROUP BY Clause

SELECT SUM(column_name) FROM table_name WHERE CONDITION GROUP BY column_name;

- SQL COUNT Clause

SELECT COUNT(column_name) FROM table_name WHERE CONDITION;

TABLE

- SQL DESC Statement

DESC table_name;

- SQL TRUNCATE TABLE Statement

TRUNCATE TABLE table_name;

- SQL ALTER TABLE Statement

ALTER TABLE table_name {ADD|DROP|MODIFY} column_name {data_ype};

- SQL ALTER TABLE Statement (Rename)

ALTER TABLE table_name RENAME TO new_table_name;

- SQL UPDATE Statement

UPDATE table_name SET column1 = value1, [WHERE CONDITION];

SQL DELETE Statement

DELETE FROM table_name WHERE {CONDITION};

TABLE

- SQL TOP Statement

```
SELECT TOP 3 * FROM CUSTOMERS;
```

- SQL TRUNCATE TABLE Statement

```
SELECT * FROM CUSTOMERS LIMIT 3;
```

- SQL ROWNUM Statement

```
SELECT * FROM CUSTOMERS WHERE ROWNUM <= 3;
```

- SQL CASE Statement (Rename)

```
SELECT * FROM CUSTOMERS
  ORDER BY (CASE ADDRESS
  WHEN 'DELHI'    THEN 1
  WHEN 'BHOPAL'  THEN 2
  WHEN 'KOTA'     THEN 3
  WHEN 'AHMEDABAD' THEN 4
    WHEN 'MP'  THEN 5
    ELSE 100 END) ASC, ADDRESS DESC;
```

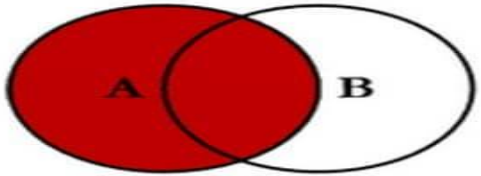
- SQL - Constraints
- SQL - Using Joins
- SQL - Unions Clause
- SQL - NULL Values
- SQL - Alias Syntax
- SQL - Indexes
- SQL - Alter Command
- SQL - Truncate Table
- SQL - Using Views
- SQL - Having Clause
- SQL - Transactions
- SQL - Transactions
- SQL - Wildcards
- SQL - Date Functions
- SQL - Temporary Tables
- SQL - Clone Tables
- SQL - Sub Queries
- SQL - Using Sequences
- SQL - Handling Duplicates
- SQL - Injection

constraints

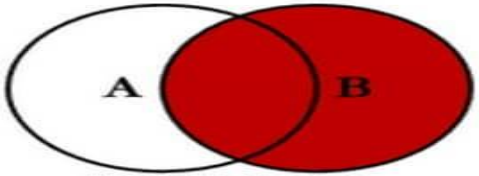
- NOT NULL Constraint − Ensures that a column cannot have NULL value.
- DEFAULT Constraint − Provides a default value for a column when none is specified.
- UNIQUE Constraint − Ensures that all values in a column are different.(null allowed,ph.no)
- PRIMARY Key − Uniquely identifies each row/record in a database table.(no null,duplicate)
- FOREIGN Key − Uniquely identifies a row/record in any of the given database table(join id is forgien key).
- CHECK Constraint − The CHECK constraint ensures that all the values in a column satisfies certain conditions.
- INDEX − Used to create and retrieve data from the database very quickly.

```
ALTER TABLE EMPLOYEES DROP CONSTRAINT EMPLOYEES_PK;
```

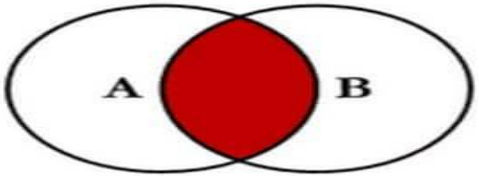
SQL JOINS



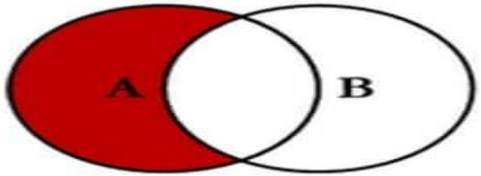

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```



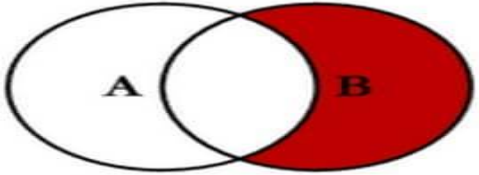

```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
```



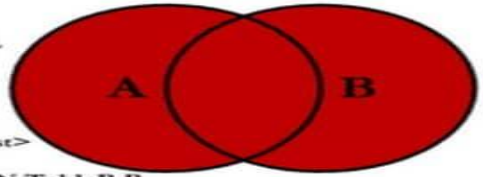

```
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
```



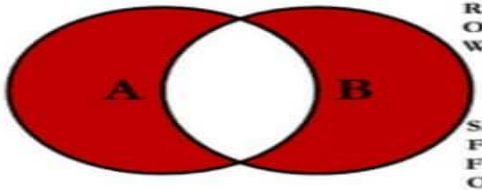

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
```




```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
```




```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
```




```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
OR B.Key IS NULL
```

joints

- INNER JOIN − returns rows when there is a match in both tables.
- LEFT JOIN − returns all rows from the left table, even if there are no matches in the right table.
- RIGHT JOIN − returns all rows from the right table, even if there are no matches in the left table.
- FULL JOIN − returns rows when there is a match in one of the tables.
- SELF JOIN − is used to join a table to itself as if the table were two tables, temporarily renaming at least one table in the SQL statement.
- CARTESIAN JOIN − returns the Cartesian product of the sets of records from the two or more joined tables.

```
ALTER TABLE EMPLOYEES DROP CONSTRAINT EMPLOYEES_PK;
```

```
  select hobbies.name,persons.name from persons
 join hobbies
 on persons.id=hobbies.person_id
 where persons.name="Bobby McBobbyFace"
select customers.name,customers.email,orders.item,orders.price
 from customers
 left OUTER join orders
 on customers.id=orders.customer_id ;
select customers.name,customers.email,sum(orders.price) as total_amount
 from customers
 left OUTER join orders
 on customers.id=orders.customer_id  GROUP by customers.name order by total_amount desc;

select movies.title,sequel.title
FROM movies
join movies sequel
on movies.sequel_id=sequel.id;

 select persons.fullname,a.fullname from friends
  join persons
  on friends.person1_id=persons.id
  join persons a
  on  friends.person2_id=a.id;
```

Unions exists clause

There are two other clauses (i.e., operators), which are like the UNION clause.

- SQL INTERSECT Clause − This is used to combine two SELECT statements, but returns rows only from the first SELECT statement that are identical to a row in the second SELECT statement.
- SQL EXCEPT Clause − This combines two SELECT statements and returns rows from the first SELECT statement that are not returned by the second SELECT statement.
- The SQL UNION clause/operator is used to combine the results of two or more SELECT statements without returning any duplicate rows.
- The UNION ALL operator is used to combine the results of two SELECT statements including duplicate rows.
- The EXISTS operator is used to test for the existence of any record in a subquery.

```
SELECT column1 [, column2 ]FROM table1 [, table2 ] [WHERE condition]
UNION ALL
SELECT column1 [, column2 ] FROM table1 [, table2 ] [WHERE condition]
```

```
SELECT column_name(s)
FROM table_name
WHERE EXISTS
(SELECT column_name FROM table_name WHERE condition);
```

NULL VALUES ANY &ALL

The SQL NULL is the term used to represent a missing value. A NULL value in a table is a value in a field that appears to be blank

```
SQL> CREATE TABLE CUSTOMERS(
   ID   INT              NOT NULL,
   NAME VARCHAR (20)     NOT NULL,(primary key does not accept null,unique allows)
   AGE  INT              NOT NULL,
   ADDRESS  CHAR (25) ,
   SALARY   DECIMAL (18, 2),
   PRIMARY KEY (ID)
);
SELECT  ID, NAME, AGE, ADDRESS, SALARY
   FROM CUSTOMERS
   WHERE SALARY IS NOT NULL;
```

ANY AND ALL

```
SELECT ProductName
FROM Products
WHERE ProductID = ANY (SELECT ProductID FROM OrderDetails WHERE
Quantity = 10);
```

ALIAS

You can rename a table or a column temporarily by giving another name known as Alias.

The basic syntax of a table alias is as follows.

```
SELECT column1, column2....
FROM table_name AS alias_name
WHERE [condition];
```

The basic syntax of a column alias is as follows.

```
SELECT column_name AS alias_name
FROM table_name
 WHERE [condition];
```

indexes

Indexes are special lookup tables that the database search engine can use to speed up data retrieval. Simply put, an index is a pointer to data in a table. An index in a database is very similar to an index in the back of a book.
but it slows down data input, with the UPDATE and the INSERT statements.

CREATE INDEX

```
CREATE INDEX index_name ON table_name;
```

Single-Column Indexes

```
CREATE INDEX index_name ON table_name (column_name);
```

Unique Indexes

```
CREATE UNIQUE INDEX index_name  on table_name (column_name);
```

Composite Indexes

```
CREATE INDEX index_name  on table_name (column1, column2);
```

DROP INDEX

```
DROP INDEX index_name;
```

ALTER CONSTRAINTS

add a New Column in an existing table is as follows.

```
ALTER TABLE table_name ADD column_name datatype;
```

DROP COLUMN in an existing table is as follows.

```
ALTER TABLE table_name DROP COLUMN column_name;
```

change the DATA TYPE of a column in a table is as follows.

```
ALTER TABLE table_name MODIFY COLUMN column_name datatype;
```

add a NOT NULL constraint to a column in a table is as follows.

```
ALTER TABLE table_name MODIFY column_name datatype NOT NULL;
```

ADD UNIQUE CONSTRAINT to a table is as follows.

```
ALTER TABLE table_name  ADD CONSTRAINT MyUniqueConstraint UNIQUE(column1, column2...);
```

ADD CHECK CONSTRAINT to a table is as follows.

```
ALTER TABLE table_name  ADD CONSTRAINT MyUniqueConstraint CHECK (CONDITION);
```

ADD PRIMARY KEY constraint to a table is as follows.

```
ALTER TABLE table_name  ADD CONSTRAINT MyPrimaryKey  PRIMARY KEY (column1, column2...);
```

ALTER CONSTRAINTS

The basic syntax of an ALTER TABLE command to DROP CONSTRAINT from a table is as follows.

```
ALTER TABLE table_name  DROP CONSTRAINT MyUniqueConstraint;
```

```
ALTER TABLE table_name DROP INDEX MyUniqueConstraint;
```

The basic syntax of an ALTER TABLE command to DROP PRIMARY KEY constraint from a table is as follows.

```
ALTER TABLE table_name  DROP CONSTRAINT MyPrimaryKey;
```

```
ALTER TABLE table_name  DROP PRIMARY KEY;
```

TRUNCATE

The SQL TRUNCATE TABLE command is used to delete complete data from an existing table.

You can also use DROP TABLE command to delete complete table but it would remove complete table structure form the database and you would need to re-create this table once again if you wish you store some data.

```
TRUNCATE TABLE  table_name;
```

USING VIEW

A view is nothing more than a SQL statement that is stored in the database with an associated name. A view is actually a composition of a table in the form of a predefined SQL query.

A view can contain all rows of a table or select rows from a table. A view can be created from one or many tables which depends on the written SQL query to create a view.

```
CREATE VIEW view_name AS
SELECT column1, column2.....
FROM table_name
WHERE [condition];
```

```
CREATE VIEW CUSTOMERS_VIEW AS
SELECT name, age
FROM  CUSTOMERS
WHERE age IS NOT NULL
WITH CHECK OPTION;
```

```
DROP VIEW view_name;
```

If they do not satisfy the condition(s) WITH CHECK OPTION, the UPDATE or INSERT returns an error.

TRANSACTION

Transactional control commands are only used with the DML Commands such as - INSERT, UPDATE and DELETE only.

Transaction Control

The following commands are used to control transactions.

- COMMIT − to save the changes.
- ROLLBACK − to roll back the changes.
- SAVEPOINT − creates points within the groups of transactions in which to ROLLBACK.
- SET TRANSACTION − Places a name on a transaction.

SET TRANSACTION [READ WRITE | READ ONLY];

WILD CARD

LIKE OPERATOR

The percent sign (%)

TEMPORARY TABLES

The temporary tables could be very useful in some cases to keep temporary data. The most important thing that should be known for temporary tables is that they will be deleted when the current client session terminates.

```
CREATE TEMPORARY TABLE SALESSUMMARY (
   product_name VARCHAR(50) NOT NULL
   total_sales DECIMAL(12,2) NOT NULL DEFAULT 0.00
   avg_unit_price DECIMAL(7,2) NOT NULL DEFAULT 0.00
   total_units_sold INT UNSIGNED NOT NULL DEFAULT 0
);
```

```
DROP TABLE SALESSUMMARY
```

CLONE TABLES

```
     Table: TUTORIALS_TBL
Create Table: CREATE TABLE 'TUTORIALS_TBL' (
  'tutorial_id' int(11) NOT NULL auto_increment,
  'tutorial_title' varchar(100) NOT NULL default '',
  'tutorial_author' varchar(40) NOT NULL default '',
  'submission_date' date default NULL,
  PRIMARY KEY  ('tutorial_id'),
  UNIQUE KEY 'AUTHOR_INDEX' ('tutorial_author')
) TYPE = MyISAM
```

```
SQL> CREATE TABLE `CLONE_TBL` (
  -> 'tutorial_id' int(11) NOT NULL auto_increment,
  -> 'tutorial_title' varchar(100) NOT NULL default
'',
  -> 'tutorial_author' varchar(40) NOT NULL default
'',
  -> 'submission_date' date default NULL,
  -> PRIMARY KEY  (`tutorial_id'),
  -> UNIQUE KEY 'AUTHOR_INDEX' ('tutorial_author')
-> ) TYPE = MyISAM;
```

```
SQL> INSERT INTO CLONE_TBL (tutorial_id,
   ->                       tutorial_title,
   ->                       tutorial_author
   ->                       submission_date
   -> SELECT tutorial_id,tutorial_title,
   ->        tutorial_author,submission_date
   -> FROM TUTORIALS_TBL;
```

```
select first_name || last_name as “Name” from students;
```

```
SELECT * FROM sales;

SELECT PRODUCT,COUNT(PRODUCT) FROM sales GROUP BY PRODUCT;

SELECT payment_type,COUNT(payment_type),product FROM sales GROUP BY payment_type,product;

SELECT DISTINCT product FROM SALES ORDER BY PRODUCT DESC;

SELECT DISTINCT city,count(city) as city_orders FROM sales  group by city having city_orders BETWEEN 2 and 3 order by city_orders desc;

SELECT country,count(country)as country_orders from sales group by country having country_orders>7 order by country DESC  ;

SELECT country,count()as country_orders from sales group by country having country_orders>7 order by country DESC  ;

select country ,sum(price) as country_sales from sales group by price;

select country,state ,sum(price)as state_sale from sales group by state order by country;

select city,sum(price)as city_sale,country from sales group by city order by city_sale desc;

select product,sum(price) from sales group by product;
```

```
PRAGMA TABLE_INFO(sales);

select DISTINCT product from sales;

SELECT  price from sales  limit 5;

SELECT  price from sales  rownum 5;

select country,sum(price) from sales where product like ("Chair") group by country order by country DESC  ;

select country,sum(price) from sales where product  not like ("Chair") group by country order by country DESC  ;

select * from sales limit 2;

 update sales set price=500  where id=1 and product="Chair";


select * from sales  order by sales.ID DESC limit 2;

SELECT * FROM astronauts WHERE year IS  NULL;

SELECT * FROM astronauts WHERE year IS  not NULL;

alter table astronauts rename COLUMN year to DOBYEAR

ALTER TABLE astronauts ADD COLUMN happy NOT NULL DEFAULT 'yes';
```

What is SQL?

Structured Query Language or SQL is a standard Database language which is used to create, maintain and retrieve the relational database

Types of SQL Commands

- DDL - Define the schema of database or its objects (like tables and indexes) (Ex - CREATE, ALTER, DROP)
- DML - Manipulate and Select data in the database (Ex – SELECT, INSERT)
- DCL - Rights, permissions and other controls of the database system (Ex – GRANT, REVOKE)

Most Common Datatypes

- int(10)
- varchar(255)
- text
- TIMESTAMP
- ENUM ('Choice1', 'Choice2', ...)

Importing and Exporting data from CSV

Importing data from CSV - LOAD

```
DATA LOCAL INFILE <full_file_path>
INTO TABLE <tableName>
COLUMNS TERMINATED BY ','
OPTIONALLY ENCLOSED BY '"' ESCAPED BY '"'
LINES TERMINATED BY '\n'
 IGNORE 1 LINES;
```

Exporting data to CSV - SELECT *

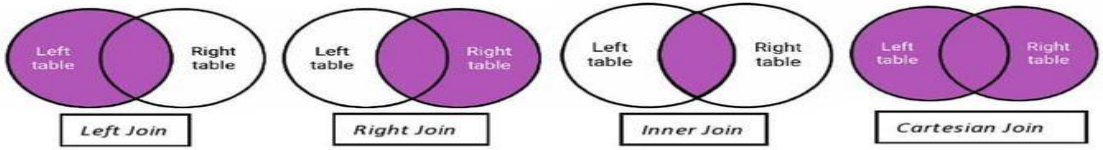
```
INTO OUTFILE <outputFilePath>
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\n' FROM <tableName>;
```

Creating a table Syntax

```
CREATE TABLE <tableName> (

        <fieldname1><fieldtype1> (NULL/NNOT NULL) ,
<fieldname2><fieldtype2>
);
```

Joins



Inserting data in a table

INSERT INTO<*tableName*> (*column1*, *column2*, *column3*, ...)
VALUES (*value1*, *value2*, *value3*, ...);

Basic Query Syntax

```
SELECT f(col1), g(col2),... from table1
---filter the rows
WHERE col2=1 and col5=4
---Aggregate the data
GROUPBY ..
--- Filter the results
HAVING h(col4)>=<...
--- Sort the results
ORDER BY col2
```

f, g, h are aggregation functions like
COUNT(*), COUNT(DISTINCT), SUM(), STDDEV() etc

Other useful Keywords which work with SELECT
DISTINCT
LIKE
BETWEEN
IN

What is an index?

- Data structure that improves the speed of operations in a table.
- Indexes can be created using one or more columns
- Indexing a column improves search but increases the time for insert and update

Creating an Index

create index <index_name> on <table> (<Column to Index>)

SQL Cheat-Sheet

Source: Analytics Vidhya, Via: Machine Learning India - @ml.india

SQL AGGREGATE FUNCTIONS

AVG returns the average of a list

COUNT returns the number of elements of a list

SUM returns the total of a list

MAX returns the maximum value in a list

MIN returns the minimum value in a list

MANAGING TRIGGERS

```
CREATE OR MODIFY TRIGGER
trigger_name
WHEN EVENT
ON table_name TRIGGER_TYPE
EXECUTE stored_procedure;
```

Create ormodifya trigger

WHEN

- **BEFORE** –invokebefore the event occurs
- **AFTER** –invokeafter the event occurs

EVENT

- **INSERT** –invokefor INSERT
- **UPDATE** –invokefor UPDATE
- **DELETE** –invokefor DELETE

TRIGGER_TYPE

- **FOR EACH ROW**
- **FOR EACH STATEMENT**

```
CREATE TRIGGER
before_insert_person
BEFORE INSERT
ON person FOR EACH ROW
EXECUTE stored_procedure;
```

Create a trigger invoked before a new row is inserted into the person table

```
DROP TRIGGER trigger_name;
```

Delete a specific trigger

MANAGING VIEWS

```
CREATE VIEW v(c1,c2)
AS
SELECT c1, c2
FROM t;
```
Create a new view that consists of c1 and c2

```
CREATE VIEW v(c1,c2)
AS
SELECT c1, c2
FROM t;
WITH [CASCADED | LOCAL] CHECK
OPTION;
```
Create a new view with check option

```
CREATE RECURSIVE VIEW v
AS
select-statement--anchor part
UNION [ALL]
select-statement;--recursive part
```
Create a recursive view

```
CREATE TEMPORARY VIEW v
AS
SELECT c1, c2
FROM t;
```
Create a temporary view

```
DROP VIEW view_name;
```
Delete a view

MANAGING INDEXES

```
CREATE INDEX idx_name
ON t(c1,c2);
```
Create an index on c1 and c2 of the table t

```
CREATE UNIQUE INDEX idx_name
ON t(c3,c4);
```
Create a unique index on c3, c4 of the table t

```
DROP INDEX idx_name;
```
Drop an index

USING SQL CONSTRAINTS

```
CREATE TABLE t(
        c1 INT, c2 INT, c3 VARCHAR,
        PRIMARY KEY (c1,c2)
);
```
Set c1 and c2 as a primary key

```
CREATE TABLE t1(
        c1 INT PRIMARY KEY,
        c2 INT,
        FOREIGN KEY (c2)
        REFERENCES     t2(c2)
        );
```
Set c2 column as a foreign key

```
CREATE TABLE t(
        c1 INT, c1 INT,
        UNIQUE(c2,c3)
);
```
Make the values in c1 and c2 unique

```
CREATE TABLE t(
        c1 INT, c2 INT,
        CHECK(c1> 0 AND c1 >= c2)
);
```
Ensure c1 > 0 and values in c1 >= c2

```
CREATE TABLE t(
        c1 INT PRIMARY KEY,
        c2 VARCHAR NOT NULL
);
```
Set values in c2 column not NULL

MODIFYING DATA

```
INSERT INTO t (column_list)
VALUES (value_list);
```
Insert one row into atable

```
INSERT INTO t(column_list)
VALUES (value_list),
(value_list), ....;
```
Insert multiple rows into a table

```
INSERT INTO t1(column_list)
SELECT column_list
FROM t2;
```
Insert rows from t2 into t1

```
UPDATE t
SET c1= new_value;
```
Update new value in the column c1 for all rows

```
UPDATE t
SET c1 = new_value,
c2 = new_value
WHERE condition;
```
Update values in the column c1, c2that match the condition

```
DELETE FROM t;
```
Delete all data in a table

```
DELETE FROM t
WHERE condition;
```
Delete subset of rows in a table

USING SQL OPERATORS

```
SELECT c1, c2 FROM t1
UNION [ALL]
SELECT c1, c2 FROM t2;
```
Combine rows from two queries

```
SELECT c1, c2 FROM t1
INTERSECT
SELECT c1, c2 FROM t2;
```
Return the intersection of two queries

```
SELECT c1, c2 FROM t1
MINUS
SELECT c1, c2 FROM t2;
```
Subtract a result set from another result set

```
SELECT c1, c2 FROM t1
WHERE c1 [NOT] LIKE pattern;
```
Query rows using pattern matching %, _

```
SELECT c1, c2 FROM t
WHERE c1 [NOT] IN value_list;
```
Query rows in a list

```
SELECT c1, c2 FROM t
WHERE c1 BETWEEN low AND high;
```
Query rows between two values

```
SELECT c1, c2 FROM t
WHERE c1 IS [NOT] NULL;
```
Check if values in a table is NULL or not

MANAGING TABLES

```
CREATE TABLE t (
    id INT PRIMARY KEY,
    name VARCHAR NOT NULL,
    price INT DEFAULT 0
);
```
Create a new table with three columns

```
DROP TABLE t ;
```
Delete the table from the database

```
ALTER TABLE t ADD column;
```
Add a new column to the table

```
ALTER TABLE t DROP COLUMN c ;
```
Drop column c from the table

```
ALTER TABLE t ADD constraint;
```
Add a constraint

```
ALTER TABLE t DROP constraint;
```
Drop a constraint

```
ALTER TABLE t1 RENAME TO t2;
```
Rename a table from t1 to t2

```
ALTER TABLE t1 RENAME c1 TO c2;
```
Rename column c1 to c2

```
TRUNCATE TABLE t;
```
Remove all data in a table

QUERYING DATA FROM A TABLE

```
SELECT c1, c2 FROM t;
```
Query data in columns c1, c2 from a table

```
SELECT * FROM t;
```
Query all rows and columns from a table

```
SELECT c1, c2 FROM t
WHERE condition;
```
Query data and filter rows with a condition

```
SELECT DISTINCT c1 FROM t
WHERE condition;
```
Query distinct rows from a table

```
SELECT c1, c2 FROM t
ORDER BY c1 ASC [DESC];
```
Sort the result set in ascending or descending order

```
SELECT c1, c2 FROM t
ORDER BY c1
LIMIT n OFFSET offset;
```
Skip *offset* of rows and return the next n rows

```
SELECT c1, aggregate(c2)
FROM t
GROUP BY c1;
```
Group rows using an aggregate function

```
SELECT c1, aggregate(c2)
FROM t
GROUP BY c1
HAVING condition;
```
Filter groups using HAVING clause

QUERYING FROM MULTIPLE TABLES

```
SELECT c1, c2
FROM t1
INNER JOIN t2 ON condition;
```
Inner join t1 and t2

```
SELECT c1, c2
FROM t1
LEFT JOIN t2 ON condition;
```
Left join t1 and t1

```
SELECT c1, c2
FROM t1
RIGHT JOIN t2 ON condition;
```
Right join t1 and t2

```
SELECT c1, c2
FROM t1
FULL OUTER JOIN t2 ON condition;
```
Perform full outer join

```
SELECT c1, c2
FROM t1
CROSS JOIN t2;
```
Produce a Cartesian product of rows in tables

```
SELECT c1, c2
FROM t1, t2;
```
Another way to perform cross join

```
SELECT c1, c2
FROM t1 A
INNER JOIN t2 B ON condition;
```
Join t1 to itself using INNER JOIN clause