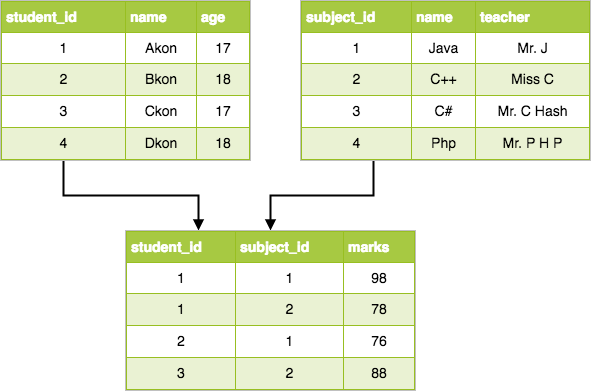
SQL for Data Science

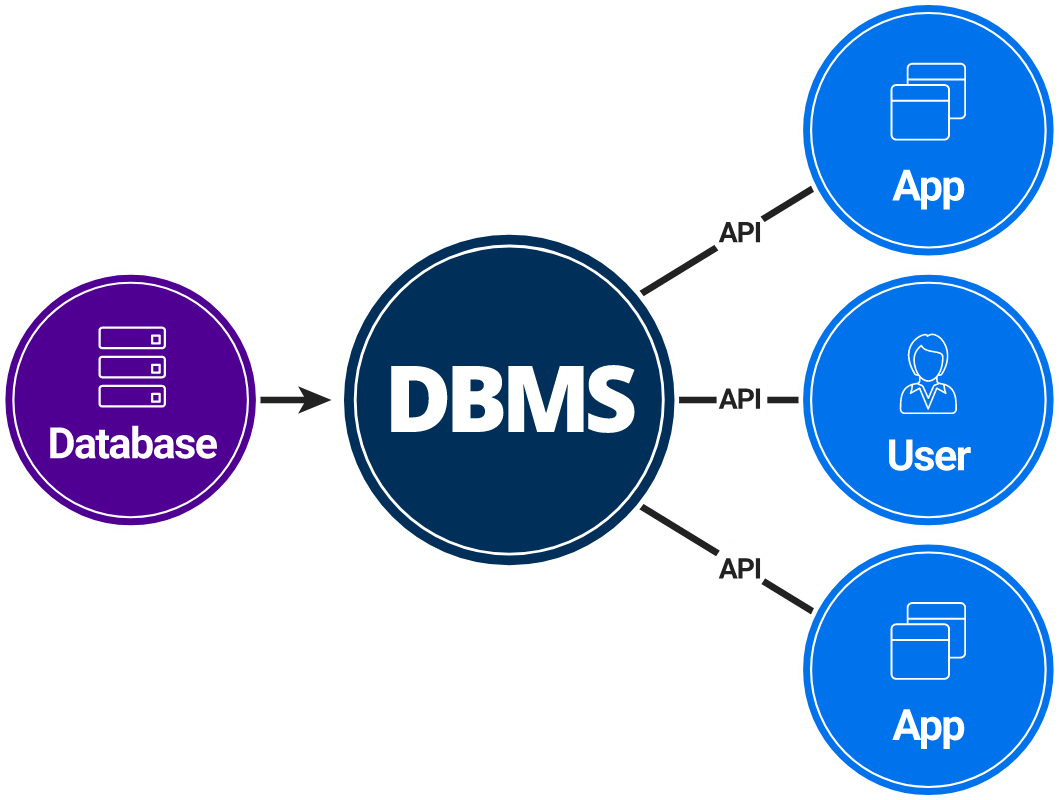
**Contents:**

1. Introduction to Databases
2. Database Management System
3. Normalization of Database
4. SQL
5. Data Definition Language
6. Data Manipulation Language
7. Data Query Language
8. Table Constraints

# Introduction to Databases:

* A database is a collection of related data items, which are linked and structured so that the data can be accessed in a number of ways.
* Within a database, related data are grouped into tables, each of which consists of rows (also called tuples) and columns, like a spreadsheet.





# Database Management System:

* Database management system is software which is used to manage the database. For example: [MySQL](https://www.javatpoint.com/mysql-tutorial), [Oracle](https://www.javatpoint.com/oracle-tutorial), etc. are a very popular commercial database which is used in different applications.
* DBMS provides an interface to perform various operations like database creation, storing data in it, updating data, creating a table in the database and a lot more.

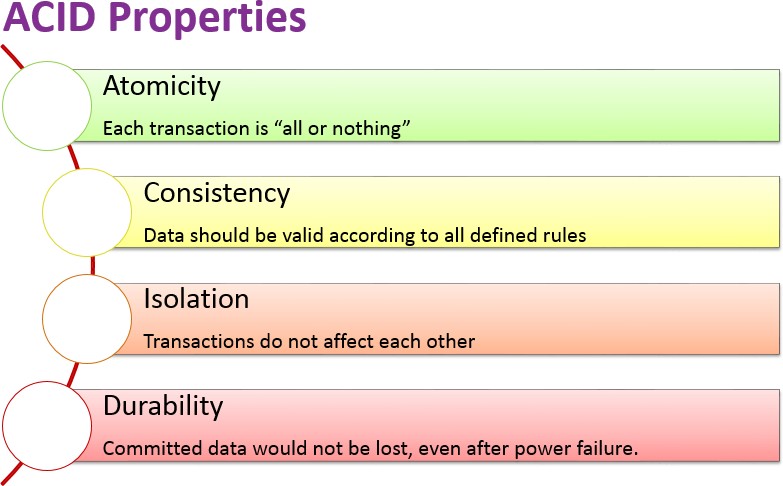
**Characteristics of DBMS**

* It can provide a clear and logical view of the process that manipulates data.
* It contains ACID properties which maintain data in a healthy state in case of failure.
* It is used to support manipulation and processing of data.
* It is used to provide security of data.

### ACID Properties:

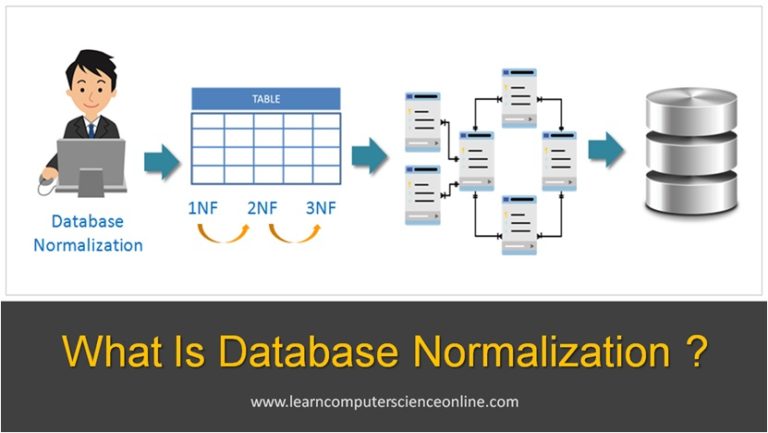
A database's contents can be accessed and possibly modified as part of a single logical unit of work known as a transaction. Read and write operations are used by transactions to access data.

Certain attributes are followed before and after the transaction in order to preserve consistency in a database. We refer to these as ACID characteristics.



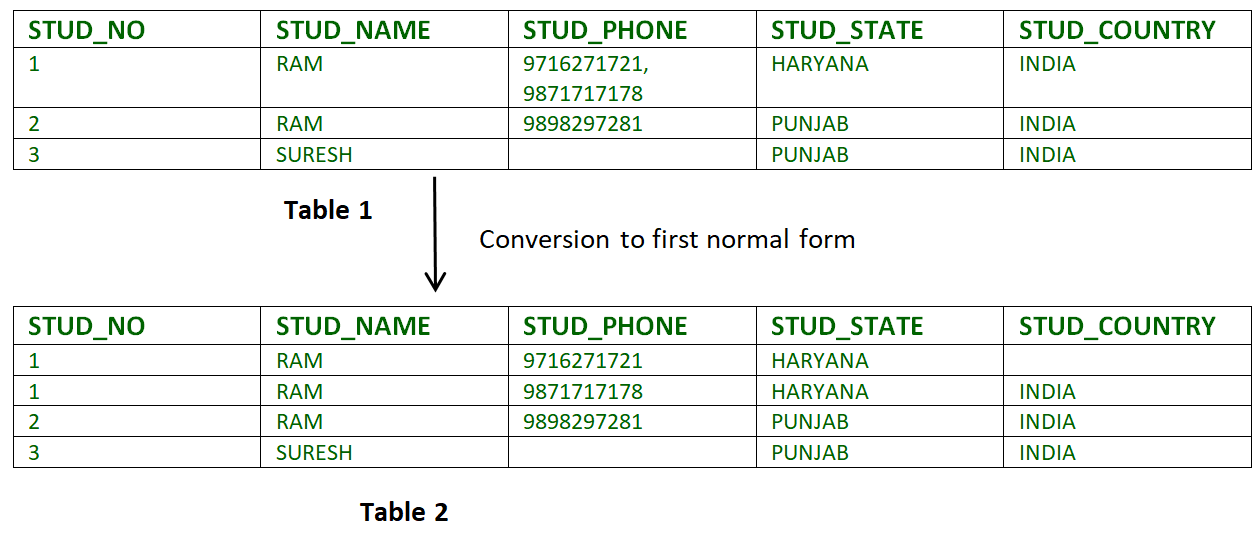
# Normalization

* Normalization is the process of organizing the data in the database.
* Normalization is used to minimize the redundancy from a relation or set of relations. It is also used to eliminate undesirable characteristics like Insertion, Update, and Deletion Anomalies.
* Normalization divides the larger table into smaller and links them using relationships.
* The normal form is used to reduce redundancy from the database table.



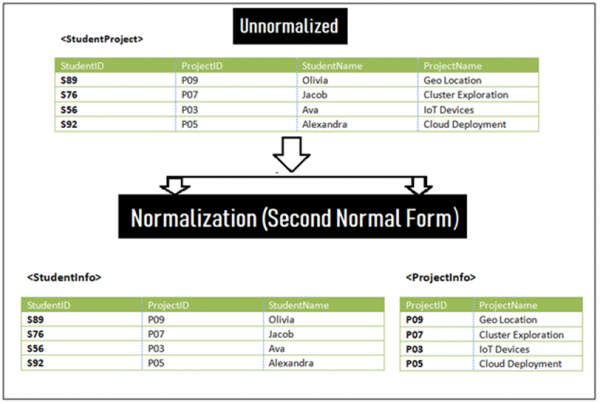
**First Normal Form (1NF)**

* A relation will be 1NF if it contains an atomic value.
* It states that an attribute of a table cannot hold multiple values. It must hold only single-valued attribute.
* First normal form disallows the multi-valued attribute, composite attribute, and their combinations.



**Second Normal Form (2NF)**

* In the 2NF, relational must be in 1NF.
* In the second normal form, all non-key attributes are fully functional dependent on the primary key.



In the above first table, we have partial dependency, let us see how −

The prime key attributes are **StudentID**and **ProjectID**.

As stated, the non-prime attributes  **StudentName**and **ProjectName**should be functionally dependent on part of a candidate key, to be Partial Dependent.

The **StudentName**can be determined by **StudentID**, which makes the relation Partial Dependent.

The **ProjectName**can be determined by **ProjectID**, which makes the relation Partial Dependent.

Therefore, the <**StudentProject**> relation violates the 2NF in Normalization and is considered a bad database design.

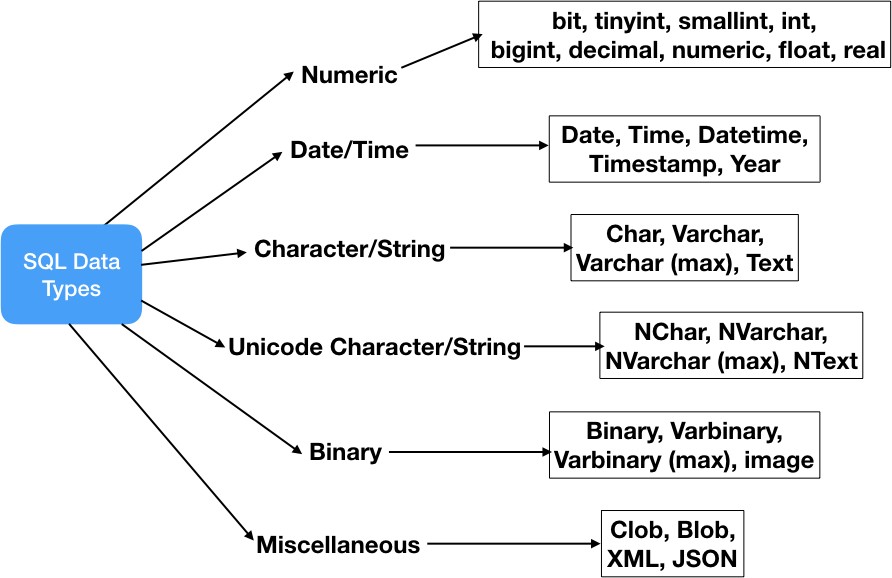
**Third Normal Form (3NF)**

A relation that is in First and Second Normal Form and in which no non- primary-key attribute is transitively dependent on the primary key, then it is in Third Normal Form (3NF).

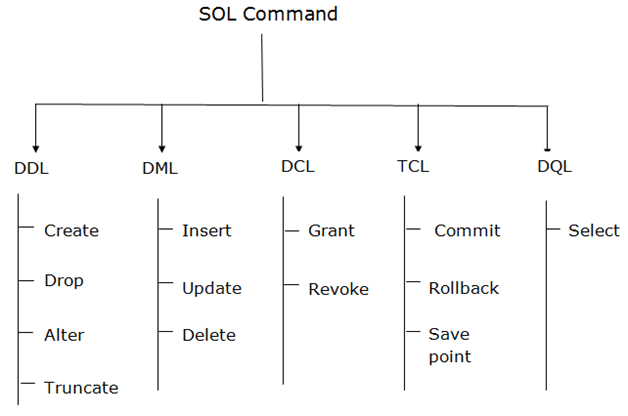
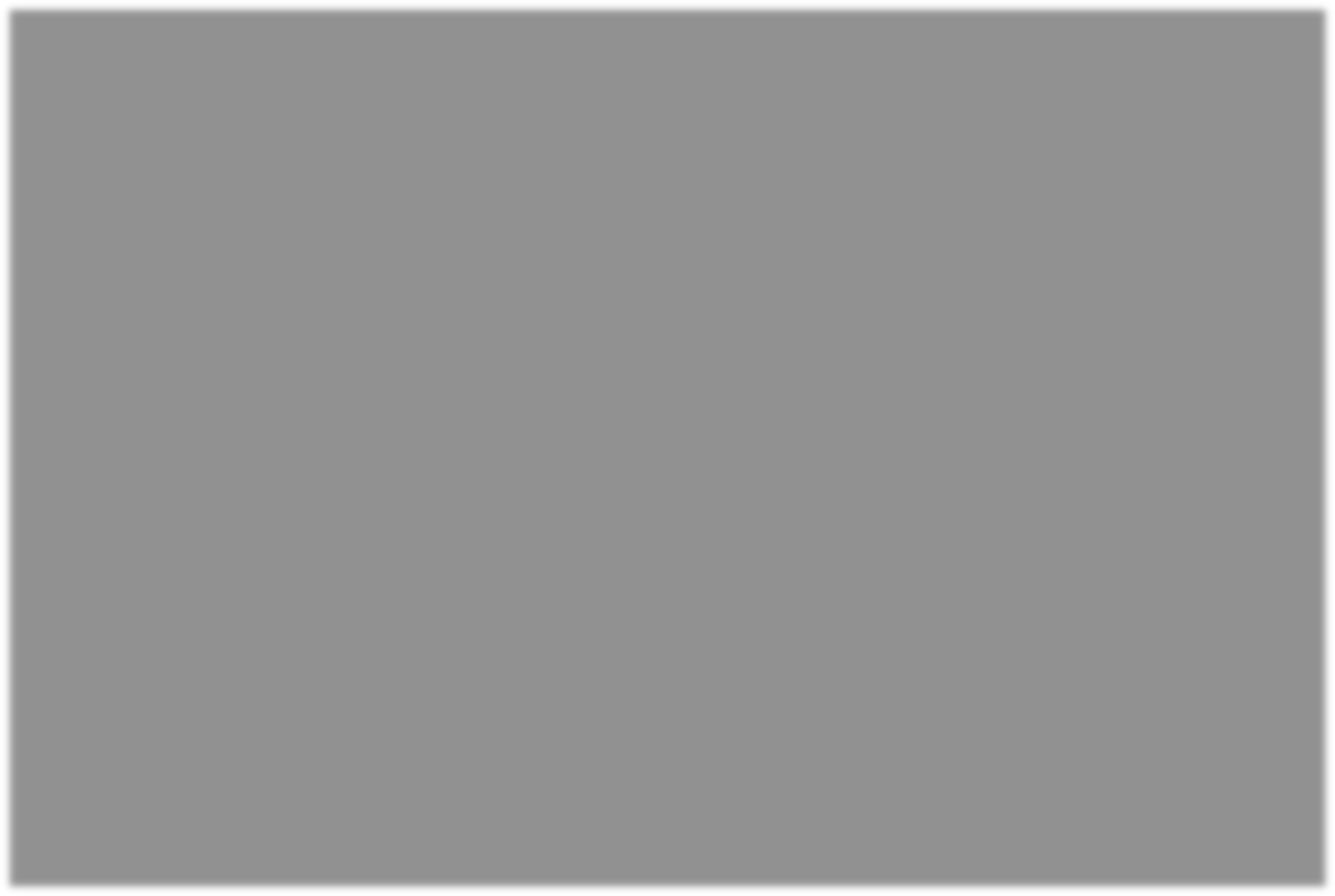
o 3NF is used to reduce the data duplication. It is also used to achieve the data integrity.

# SQL

* SQL stands for Structured Query Language. It is used for storing and managing data in relational database management system (RDMS).
* It is a standard language for Relational Database System. It enables a user to create, read, update and delete relational databases and tables.
* All the RDBMS like MySQL, Informix, Oracle, MS Access and SQL Server use SQL as their standard database language.



***Note*:** This crash course has to be completely performed in MYSQL Workbench. Select your OS type and download workbench from the given link. <https://dev.mysql.com/downloads/workbench/>





The version of MYSQL Workbench used for this assignment is 8.0.27 You can use any other stable version.

Follow[: https://youtu.be/YSOY\_NyOg40](https://youtu.be/YSOY_NyOg40)

# Data Definition Language

## Create command:

Used to create a database or a table. Syntax:

CREATE TABLE table\_name (column\_name column\_type constraints);

**Q1. Create a table ‘cars’ having columns like Sr. No., model\_name, company\_name, launch\_year, type, passengers, mileage.**

## Alter command:

An alter command modifies an existing database table. Syntax:

ALTER object type object name parameters;

**Q2. Alter the ‘cars’ table to add a column ‘rate’.**

## Drop command:

A drop command is used to delete objects such as a table, index or view.

Syntax:

DROP object type object name;

**Q3. Drop the column ‘rate’ and then the entire table.**

## Truncate command:

Similar to DROP, the TRUNCATE statement is used to quickly remove all records from a table.

Syntax:

TRUNCATE TABLE table\_name;

**Q4. Truncate the ‘cars’ table.**

# Data Manipulation Language

## Insert command:

Used to insert records in a database. Syntax:

INSERT INTO table\_name (column\_names) VALUES (records);

**Q5. Insert some suitable records in the table.**

## Update command:

Command to change or update current/existing data. Syntax:

UPDATE table\_name SET column\_name = value WHERE condition;

**Q6. Update the mileage of the ‘Hyundai Xcent’ car.**

## Delete command:

DELETE command is used to delete some or all records from the existing table.

Syntax:

DELETE FROM table\_name WHERE condition;

**Q7. Delete the last record from the table.**

# Data Query Language

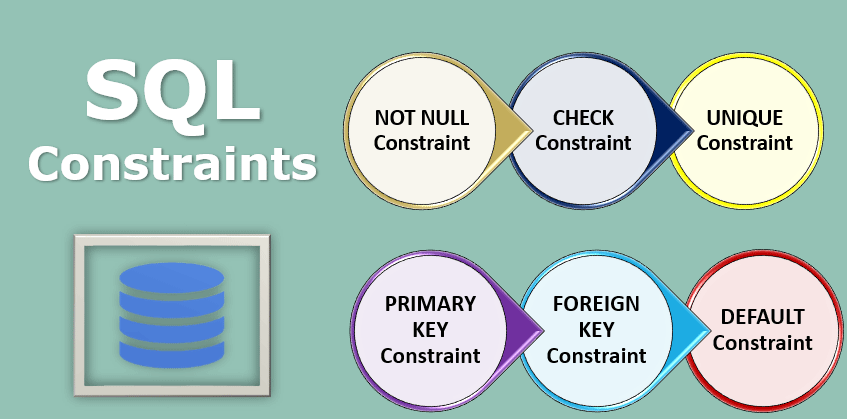
## Select command:

This command retrieves the data from the table. Syntax:

SELECT column\_names FROM table\_name;

**Q8. Display the records from cars table.**

# Table Constraints



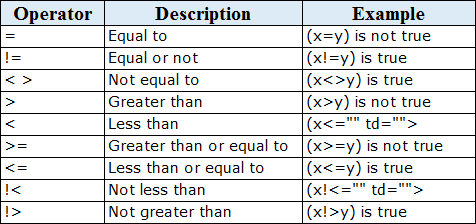
* **NOT NULL**: This constraint tells that we cannot store a null value in a column. That is, if a column is specified as NOT NULL then we will not be able to store null in this particular column any more.
* **UNIQUE**: This constraint when specified with a column, tells that all the values in the column must be unique.
* **PRIMARY KEY**: A primary key is a field which can uniquely identify each row in a table.
* **FOREIGN KEY**: A Foreign key is a field which can uniquely identify each row in another table.
* **CHECK**: This constraint helps to validate the values of a column to meet a particular condition.
* **DEFAULT**: This constraint specifies a default value for the column when no value is specified by the user.

# Where, Comparison Operators, Logical Operators

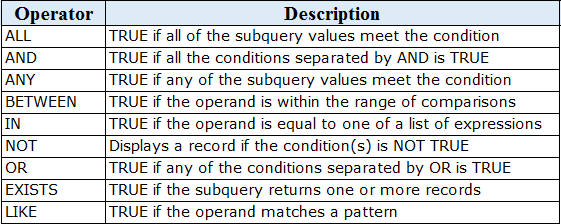
The SQL **WHERE** clause is used to specify a condition while fetching the data from a single table or by joining with multiple tables.

SELECT column1, column2, column FROM table\_name WHERE [condition]

**Comparison Operators:**



**Logical Operators:**



**Q9. Return the records of cars of company ‘Maruti Suzuki’.**

**Q10. Retrieve the car names having mileage greater than 75.**

**Q11. Get the car names having company name ‘Hyundai’ and mileage greater than 75.**

**Q12. Show the car names of the company ‘Renault’ or ‘Datsun’.**

**Q13. Display car names launched in the years between 2012 and 2016.**

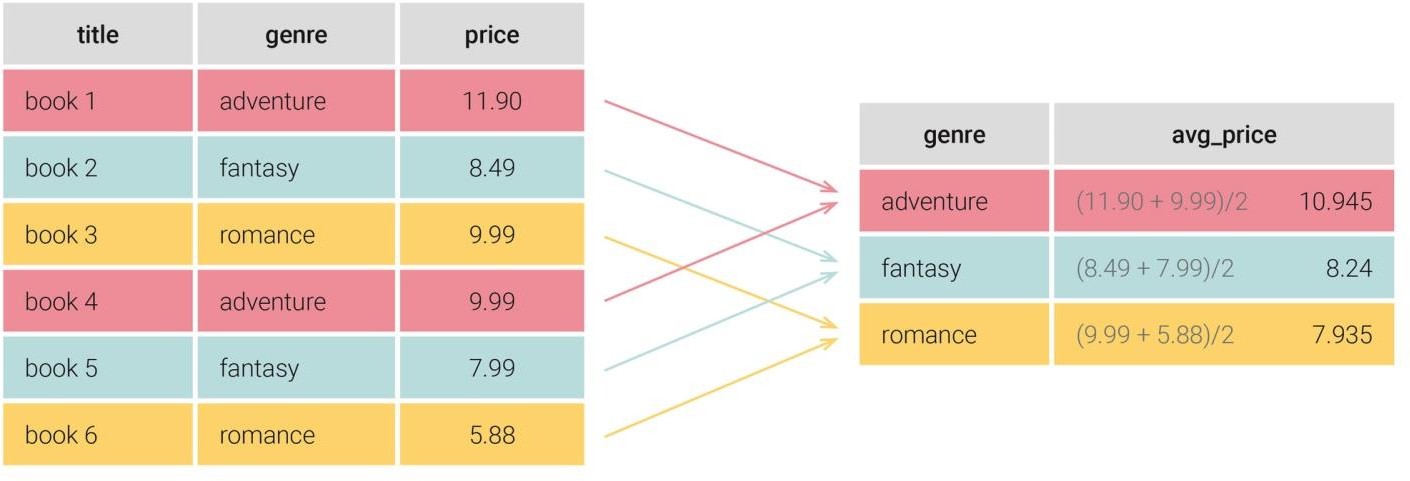
# Group By, Having, Limit, Order By

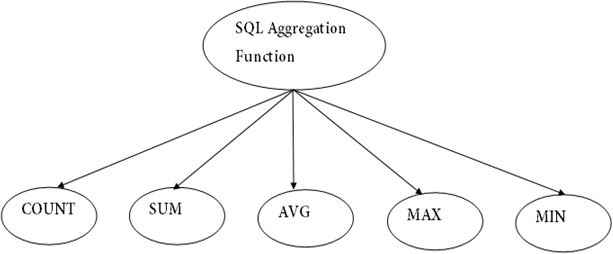
The **GROUP BY** statement groups rows that have the same values into summary rows.

**Syntax:**

SELECT column\_name(s) FROM table\_name WHERE condition

GROUP BY column\_name(s);





**Q14. Find the number of cars of each type.**

**Q15. Display the average mileage of each type of car. Q16. Calculate total car mileage for each company.**

**Q17. Display the oldest model launched for every company. Q18. Display the latest model launched for every company.**

The WHERE clause cannot be used to filter the grouped rows based on a condition. We have the **HAVING** clause for this purpose. The HAVING clause always precedes the **ORDER BY** statement in the query.

SELECT column(s) FROM table\_name WHERE [condition]

GROUP BY column\_names HAVING [condition] ORDER BY column\_names;

**Q19. Display the count of car models for each company having more than 1 model in ascending order of count.**

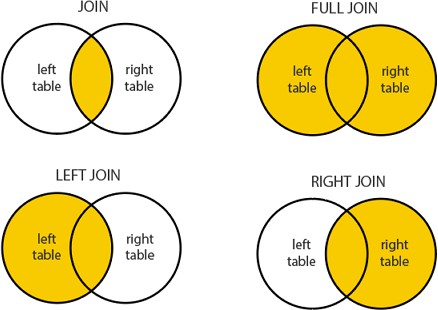
**Q20. Retrieve the car records in decreasing order of mileage.**

The **LIMIT** clause is used to set an upper **limit** on the number of records returned.

**Q21. Retrieve the top 3 records based on rate.**

# Joins

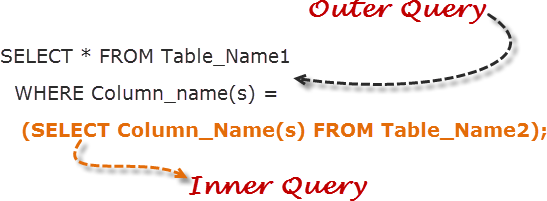
**SQL Join** statement is used to combine data or rows from two or more tables based on a common field between them. Different types of Joins are as follows:



**Q22. Display the model number, model name and buyer name for common models between two tables.**

# Nested Queries

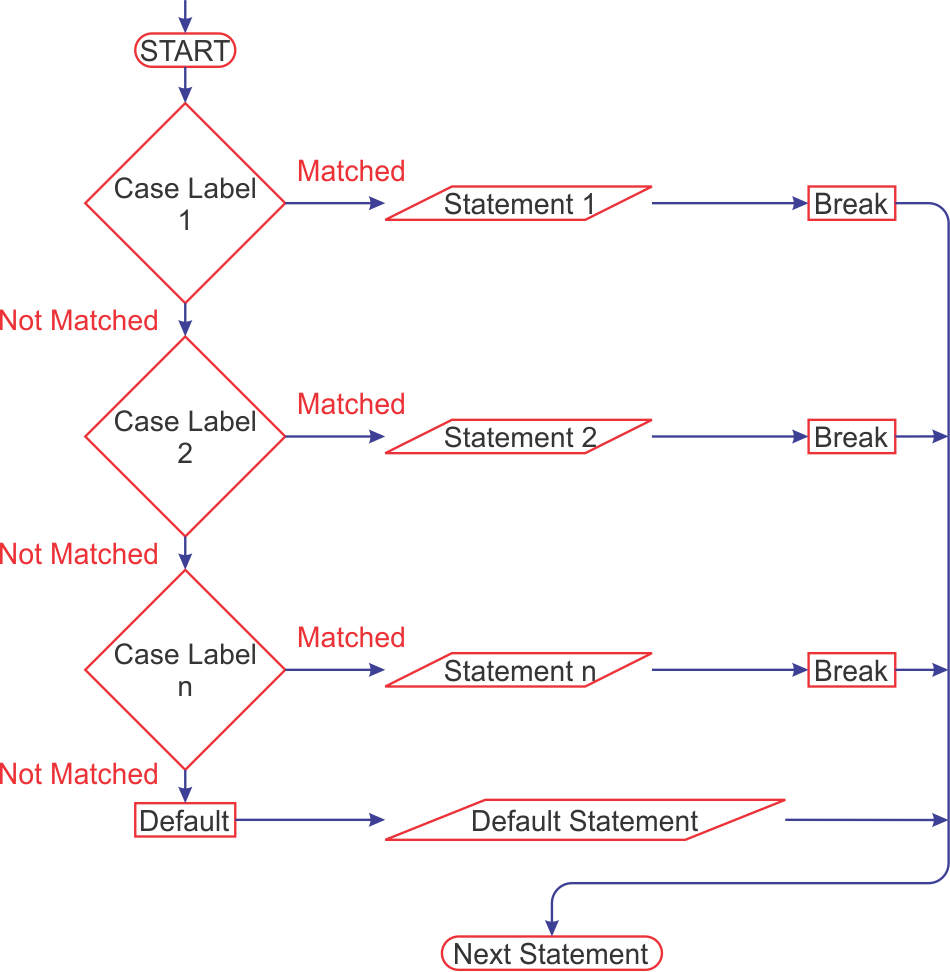
A nested query is a query that has another query embedded within it. The embedded query is called a subquery.



**Q23. Display the model number, model name of the cars that have been bought by the buyers.**

# Case When Statements, If Else Statements

The case statement in SQL returns a value on a specified condition. We can use a [Case statement](https://www.quest.com/community/blogs/b/database-management/posts/sql-case-statement-what-is-it-and-what-are-the-best-ways-to-use-it) in select queries along with Where, Order By, and Group By clause. It can be used in the Insert statement as well.



**Syntax:**

SELECT CASE Expression When expression1 Then Result1 When expression2 Then Result2

...

ELSE Result END

**Q24. Retrieve the model name and corresponding launch year in the form:**

1. ‘After 2015’ for launch years greater than 2015.
2. ‘Between 2010 and 2015’ for launch years between 2010 and 2015.
3. ‘Before 2015’ for the rest.

The [**IF**](https://dev.mysql.com/doc/refman/8.0/en/if.html) **statement** for stored programs implements a basic conditional construct.

Syntax:

IF(condition,result1,result2);

**Q25. Retrieve the model name and corresponding rate as ‘High’ for rate above 500000 and ‘Low’ for others.**

