**Introduction to Relational Databases and Tables:**

* A **database** is a **repository** of data that provides functionality for adding, modifying, and querying the data.
* **SQL** is a language used to **query** or retrieve data from **a relational database**.
* **The Relational Model** is the **most used** data model for databases because it allows for **data independence.**
* **The primary key** of a relational table **uniquely** identifies each tuple or row, **preventing duplication** of data and providing a way of defining **relationships** between tables.
* SQL statements fall into two different categories: **Data Definition Language** (DDL) statements and **Data Manipulation Language** (DML) statements.
* Data Definition Language (or DDL) statements are used to define, change, or delete database objects

**Refining your Results:**

**Simplify** a SELECT statement by using

**String patterns (**when I do not know the exact where condition of string**) =** like ‘%’,

**Ranges (**using between to identify ranges**) =** between 5 and 10, or

**Sets of values (**but values of repeated attributes in prances with IN**) =** in (,).

**Sort** **the result** set by either **ascending** (by default) or **descending** (using DESC) order, and explain how to indicate which column to use for the sorting order (using the number of column in the selected columns instead of the column name).

**Eliminate duplicates** from a result set using distinct (write after select).

**Grouping Results sets** using group by (we write the column witch has repeated values) and using having (like where but use with group by only) in addition to function apply to the selected column in group by (like count)

### Functions, Multiple Tables, and Sub-queries:

**Built in** **SQL Aggregate Functions** = sum, minimum, maximum, and average (apply to column or set of values).

**Built in** **SQL Scalar** **and** **String Functions** = round, lowercase, and uppercase (apply to just one value).

**Built in** **SQL Date** **and** **Time Functions.**

**Sub-Queries and Nested Selects,** use in

* Where Clouse to overcome some of the limitation of aggregation function (like AVG)’
* List of Columns (sub-query in place of a column) like assign AVG value to each row
* From Clouse as a data source (sub-query in place of a table).

**Working with Multiple tables**

1. **Sub-queries**
2. **cross join (implicit join) =** SELECT column\_name(s) FROM table1, table2;
3. **Join operation (inner and outer)**

Note when we add Where Clouse to **cross join** it called **inner join**

Example “SELECT column\_name(s) FROM table1, table2 WHERE table1.column\_name = table2.column\_name;”