**SEEDING:**

* SEEDING: Populating a database for the first time:
* All the SQL statements are auto commit- meaning that they will save the changes to disk as soon as they are executed.
* Sometimes we don’t want to auto commit, like, in case of running the seeding script.
* We can use the BEGIN Transaction statement to achieve this:

BEGIN TRANSACTION;

<Statement1>

<Statement2>

<Statement 3>

// At this point of time the above changes are only visible to us and no others as they are not

// Committed. To rollback these changes we can simply use:

// ROLLBACK;

COMMIT; // all the above statements will be committed now.

ORM:

* Stands for Object- Relational Mapping
* Used to perform CRUD operations with language other than SQL
* Developer is exposed to one programming language and ORM still uses SQL under the hood.

**DML operations:**

1. Four Basic operations can be performed in a database:
2. Create
3. Retrieve
4. Update
5. Delete

* A database can contain 3 types of data

1. Integer
2. Text
3. Date
4. **Create**: To add a data in a database

* INSERT INTO <Table\_Name> VALUES ( <Value1>, < Value2>….>
* Values should be inserted in the same order as we want to enter them in each columns.
* We can be a bit more specific by providing the column names as well:
* INSERT INTO <TABLE> ( <Col1> , <Col2>, …. > VALUES (<Value1>, <Value2.. >)

Inserting multiple data at the same time:

INSERT INTO <Table>

(<Col1>, <Col2>, …..)

VALUES (<Value1>, <Value 2>…. ),

(<Values1>, <Values2>…),

…

..

1. Update: Update a row in a data base

* UPDATE <Table> SET <column> = <value>
* Updating multiple columns in a single statement
* UPDATE <Table> SET <column1> = <value1> , <Column 2> = <value2>
* Updating based on conditions:
* UPDATE <Table> SET <column> = <value> WHERE <Condition>

1. Delete: To delete one or more rows in a table:

* DELETE FROM <Table>;
* DELETE FROM <Table> where <Condition>;

1. Retrieving: To retrieve a data from database:

SELECT <Column1>, <Column2> from <Table> WHERE <Condition>

**Reporting with SQL**

1. Retrieving data in a specific order: **ORDER BY**

* SELECT <columns> FROM <Table> ORDER BY <column> ASC/DESC
* By default, the sorting is ascending order. But we can use ASC or DESC for ascending or descending order, respectively.
* We can add secondary ordering as well.
* SELECT <columns> FROM <table> ORDER BY <column1> ASC , <column2> DESC

NULL Values: Empty values are treated as NULL. In SQL nothing is every equal to NULL. So we can’t use following condition:

WHERE <COL\_NAME> = NULL

We have to use:

WHERE <COL\_NAME> IS NULL

WHERE <COL\_NAME> IS NOT NULL

1. Limiting number of results: LIMIT

* SELECT <columns> FROM <Table> LIMIT <number of rows>
* LIMIT must be at the end of the SQL syntax.

1. Skipping the rows in the results : OFFSET

* This can be done using OFFSET keyword after LIMIT
* SELECT <columns> FROM <Table> LIMIT <# of rows> OFFSET <skipped rows>
* This will skip the first n rows listed after OFFSET and query the data from the remaining rows.

1. Functions: The manipulate the data returned by SQL query.

* Syntax: <function Name> (Value or column)

1. Aliasing column names to be more human readable:

* SELECT <column> AS <Readable\_Name> FROM <Table>

1. Concatenation of values or columns:

* This can be used to manipulate the data returned by sql query by combining data from different columns
* SELECT concat( <column1, column2> AS <New\_Column\_Name> FROM <Table>

1. Finding Length of text in column: LENGTH(<column>)

* SELECT <column> FROM <table> WHERE LENGTH(<column>) > <some\_length>
* SELECT LENGTH (<column>) FROM <Table>;
* Can be used in both SELECT and CONDITION section of the query

1. Changing Cases for value or columns: UPPER(value or column) , LOWER (column or Value)

* Can be used with both SELECT and CONDITION section of the query.

1. Creating excerpts from the Text: SUBSTR(<column>, begin, end)

* SELECT <column1> SUBSTR(<column2> , 1, 30) FROM <Table>
* Can be used with both SELECT and CONDITION section of the query.

1. Replacing portions of text with other text: replace(<column or value>, <target> , <replacement>)

* Can be used with both SELECT and CONDITION section of the query.

1. Counting Results: COUNT

* To count all the rows in a table
* Counting the number of results in a query.
* Counting unique entries in a table.
* Count only the counts where values are present. It does not count the Null columns.
* SELECT COUNT(\*) FROM <Table>
* DISTINCT keyword only shows rows with unique values. (where as Group by groups together the unique values so that we can use keywords like Count on them.
* SELECT COUNT(DISTINCT <column>) FROM <Table>

1. Fetching columns grouped a particular value in the table:

* For example : Shoes grouped by Running Shoes, Badminton Shoes, etc.
* SELECT <column> FROM <table> GROUP BY <column>
* SELECT <column> , COUNT (\*) FROM <table> GROUP BY <Column>

1. SUM – to sum up a numeric column

* SELECT SUM (<column>) FROM <table>
* WHERE condition filter the result set before Grouping things together. So we can’t use WHERE with GROUP By. For this we, HAVING keyword, which works with Aggregated users.
* HAVING also needs to be used before ORDER BY Keyword.

1. AVG – Average function to find the average value of a numeric column
2. MAX – find maximum value out of a numeric column.
3. MIN – find minimum value out of a numeric column.
4. ROUND(column , decimal point) – used to round of decimal values in the result

DATE AND TIME:

1. NOW() and DATE(‘now’) can be used to get today’s date

Table Joins:

* Used for combining data from one table to another.
* Using join in a query tells the database to query from two different tables together into one result set.
* JOINS can be used with one to one, one to many and many to many relationship.
* Types of Join:

1. INNER JOIN

-To get the data if the column value matches in both the table.

-Similar to intersection in Sets.

-We can inner join in more than two tables.

-The equality criteria is usually a primary key on table and foreign key on the other.

-SELECT \*

FROM <Table1> AS T1

INNER JOIN <Table2> as T2 ON t1.column = t2.column

1. OUTER JOIN:

-Get data from two tables if data is present on both the tables and the data which is not common.

-LEFT OUTER JOIN: Get data common in both table + the data present in first table only

-RIGHT OUTER JOIN: Get data common in both tables + the data unique to second table.

-FULL OUTER JOIN: Get data common in both tables + the unique data in both the tables.