CH-4 :- SQL (~~DML~~ DDL)

- SQL is case sensitive but not commands
  data

Why SQL ?

It is not only used to create structure but it is also used to query to create report from the DB.
It is a structured way.
Also used for data analysis

⇒ DDL Commands :-

i) CREATE TABLE :- Things to consider before we create our table are :-
- Type of data (decimal, int, char, pict.)
- The table name (no space, only underscore is allowed)
- What columns will make up the pri. key
- The name of columns (same as table name)

CREATE TABLE <table_name> (field1 datatype constraint, field2 datatype constraints...);

ii) ALTER TABLE :-

ALTER TABLE <table_name> ADD|DROP <column_name> [datatype];
(for ADD command)

iii) DROP TABLE :-

DROP TABLE <table_name>;

→ Delete Vs Drop :-

DELETE	DROP
It is a DML command & used when you want to remove some or all the tuples from a relation. If WHERE clause is used along with the DELETE	It is a DDL command which removes the named elements of the schema like relations, domains or constraints & you can also remove an entire schema using DROP Command

ROLLBACK :-

Rollback → Actions performed by DELETE can be rolled back as it uses buffer → Actions performed by DROP can't be rolled back because it directly works on actual data

ORDER BY :- By default it is asc.

SELECT * FROM <table_name> ORDER BY <column_name> ASC|DESC;

It is used to store data in a proper manner. All data line up in sequence

GROUP BY :- divide the tuples into groups & returns an aggregate for each groups. Always used along with aggregate func^n. Usually, it is an aggregate func^n's companion

SELECT aggregate func^n(<column_name>), ... FROM <table_name> GROUP BY <column_name>;

AS :- This func^n is used to name a func^n when we are performing an func^n aggregate func^n. for e.g:-

~~SELECT food, sum, date~~

SELECT food, SUM(sold) AS totalSold FROM FoodCart GROUP BY food;

AND, OR, NOT Operators :- Binary Operators :-

Used with WHERE clause when we want to apply conditions in two columns (AND) or two data (OR) or we want to exclude (NOT) something.

~~Eg:~~ Syntax :-

SELECT * FROM <column_name> WHERE <condition> AND <condition>;

OR ;

NOT <condition>;

e.g:-

SELECT * FROM Customers WHERE Country = 'India' AND City = 'Patna';
SELECT * FROM Customers WHERE Country = 'India' OR Country = 'Nepal';
SELECT * FROM Customers WHERE NOT Country = 'Nepal';

→ Datatype :- ~~Type of Data~~

Used to define the values that a column can contain. Every column is required to have a name & dtype in the DB table

- Binary dtypes :-

Binary :- max. length of 8000 bytes. fixed-length binary data
Varbinary :- " " " " " . Variable-length binary data
image :- " " " 2,147,483,647 bytes. Variable-length binary data.

- Approximate Number Numeric dtype :-

float :- Used to specify a floating - point value.
real :- " " " " single precision floating point no.

- Exact Numeric dtype :-

int :- Used to specify an integer value
smallint :- Used to specify small integer value
bit :- Has the no. of bits to store.
decimal :- Specifies a numeric value that can have a decimal no.
numeric :- " " " "

- Character string dtype :-

char :- Max. length of 8000 characters. fixed-length non-unicode characters
varchar :- " " " " " variable-length non-unicode characters
text :- " " " 2,147,483,647 characters. Variable-length non-unicode characters.

- Date & Time Dtypes :-

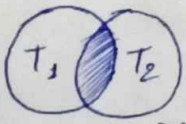
date :- Used to store the year, month, & days value
time :- " " " " hour, minute & second values.
time stamp :- year, month, day, hour, minute & the second value.

⇒ Join :- Used to joins more than one table.
When we want to extract info. from more than one table.

Types :-

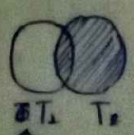
- INNER JOIN :-

One of the more frequent types of joins. finds all rows which meet the join condition.



- LEFT OUTER JOIN :-

All records b/w T1 & T2 that meet the join condition, & then any records in T1 that don't meet the join condition.



- RIGHT OUTER JOIN :-

All records b/w T1 & T2 that meet the join condition, & then any records in T2 that don't meet the join condition.

- FULL OUTER JOIN:-

All records b/w T_1 & T_2 that meet the join condition, & the combination of a left outer join & right outer join are appended onto the results

⇒ Aggregate funcⁿ :-

Used to provide summarization info. for SQL statements, which return a single value.

- COUNT(): Count all ~~rows~~ values except NULL values
- COUNT ALL(): Count all ~~values rows~~ values.
- SUM()
- MAX()
- MIN()
- AVG()

These funcⁿ operate on the multiset of values of a column of a relation, & return a value.

→ To be used in pri. key column otherwise we have to choose DISTINCT keyword ~~Command~~ to get the counting of distinct items.

HAVING clause is also used in this funcⁿ. ⇒ This clause is always ~~also~~ used after the GROUP BY clause

WHERE is used before GROUP BY & HAVING is used after GROUP BY.

→ Schema in SQL:-

It is a collection of DB obj. associated with a DB. The username of a DB is called a Schema owner (owner of logically grouped structures of data).

Schema always belong to a single DB whereas a DB can have single or multiple Schemas.

Advantages:-

- We can apply security permissions for separating & protecting DB obj. based on user access rights.
- Schemas play an imp. role in allowing the DB obj. to be organized into these logical groups
- Also helps in situations where the DB obj. name is the same.
- A single schema can be used in multiple DB.
- It also helps in adding security.

SCHEMA :-

- CREATE SCHEMA :- Create multiple tables & views & perform multiple grants in our own schema in a single transaction
- ALTER SCHEMA :- Used to rename a schema or to specify a new owner, the new owner must be a pre-existing user on DB
- DROP SCHEMA

→ real, double precision domain type :- Floating point & double precision floating point no., with machine-dependent precision

Predicate = condition

⇒ SELECT Clause :-

SQL allows duplicates in relations as well as in query results
To force the elimination of duplicates, insert the keyword DISTINCT after select.

The keyword ALL specifies that duplicates not be removed
An asterisk (*) in the select clause denotes "all attributes".

⇒ WHERE Clause :-

Specifies conditions that the result must satisfy

⇒ FROM Clause :-

Lists the relations involved in the query.
Corresponds to the Cartesian Product operation of the relational algebra.

⇒ Nested Subqueries :-

- SQL provides a mechanism for the nesting of subqueries.
- A subquery is a select-from-where expression that is nested within another query.
- A common use of subqueries is to perform tests for set membership, set comparisons & set cardinality.

"IN" Construct :-

SELECT DISTINCT <column_name> FROM <table_name>
WHERE <column_name> IN (SELECT <column_name> FROM <table_name2>);

⇒ View :-

A relation that is not of the conceptual model but is made visible to user as a "virtual relation" is called a view.

It is defined ~~used~~ using the CREATE VIEW statement which has the form

CREATE VIEW V AS <query expression>

Uses :-

Hiding some info. from some users.

⇒ BETWEEN Operator :-

It selects values within a range. The values can be numbers, text, or dates.

SELECT <column_name> FROM <table_name> WHERE <column_name> BETWEEN value 1 AND value 2;

⇒ WILDCARDS :-

Wildcards characters are used with SQL LIKE operator.

Wildcards are used to search for data within a table.

⇒ UNION :-

Used to combine the results of two or more SELECT statements without returning any duplicate rows.

SELECT <column1> [, <column2>] FROM <table1> [, <table2>]
[WHERE condition] ~~UNION~~
UNION
SELECT <column1> [, <column2>] FROM <table1> [, <table2>]
[WHERE condition]