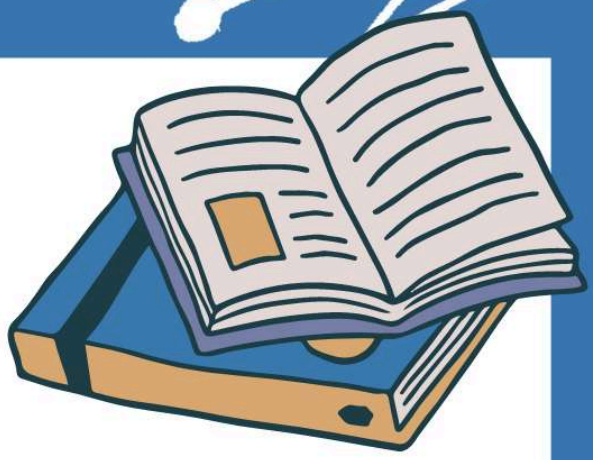


# HANDWRITTEN NOTES ON

# SQL

BY: PREKSHA MAHAJAN

__/__/____

# SQL - Complete Course

- **Database** - It is a collection of data in a format that can be easily accessed digitally.

→ A software application used to manage our DB is called database management system (DBMS).

User $\xrightarrow{SQL}$ DBMS ⟶ DB

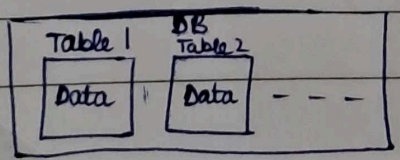
- **Types of Database** -

i) Relational (RDBMS)
→ Data stored in tables
→ SQL is used
E.g: MySQL, Oracle, PostgreSQL, Microsoft SQL Server

ii) Non-relational (NoSQL)
→ Data is not stored in tables
→ SQL is not used
mongoDB

- **SQL** - Structured Query Language is used to interact with relational databases.
It is used to perform CRUD operations:
Create
Read
Update
Delete

- **Database Structure** -




Interrelated data is stored in the form of tables.

__/__/____

- Table - It is composed of rows and columns.
  Columns :- structure / schema / design of table
  Rows :- individual data

- Queries :-

i) Creating a database
→ CREATE DATABASE db_name ;
→ CREATE DATABASE IF NOT EXISTS db_name;
ii) Deleting a database
→ DROP DATABASE db_name ;
→ DROP DATABASE IF EXISTS db_name ;
iii) Using a database
→ USE db_name ;

iv) Creating a table
→ CREATE TABLE table_name (col_name1 datatype constraint, col_name2 datatype constraint, - - -);

- SQL datatypes - They define the type of values that can be stored in a column.

1. CHAR - string (0-255), can store characters of fixed length
2. VARCHAR - string (0-255), can store characters upto given length
3. BLOB - string (0-65535), can store binary large object
4. INT - integer (-2147483648 to 2147483647)
5. TINYINT - integer (-128 to 127)
6. BIGINT - integer (-9223372036854775808 to 9223372036854775807)

__/__/____

7. BIT - can store - x bit values. x can range from 1 to 64.
8. FLOAT - Decimal number with - precision to 23 digits.
9. DOUBLE - Decimal number with - 24 to 53 digits.
10. BOOLEAN - Boolean values 0 or 1
11. DATE - date in format of YYYY-MM-DD ranging from 1000-01-01 to 9999-12-31
12. YEAR - year in 4 digits format ranging 1901 to 2155

→ Signed datatypes - They can contain -ve and +ve values

E.g: TINYINT range (-128 to 127) → Signed
TINYINT UNSIGNED range (0 to 255) → Unsigned

- Types of SQL commands -

i) DDL (Data Definition Language): create, alter, rename, truncate & drop
ii) DQL (Data Query Language): Select
iii) DML (Data manipulation Language): Insert, update & delete
iv) DCL (Data Control Language): grant & revoke permission to users
v) TCL (Transaction Control Language): Start transaction, commit & rollback

- Queries :-

__/__/____

v) See all databases on server
→ SHOW DATABASES;

vi) List of tables
→ SHOW TABLES;

vii) Delete table
→ DROP TABLE table_name;

viii) View all records of table
→ SELECT * FROM table_name;

ix) Insert data in a table
→ INSERT INTO table_name (col_name1, col_name2, ---) VALUES (col1_v1, col2_v1, ---), (col1_v2, col2_v2, ---), (---);

- Keys -

1. Primary Key - It is a column / set of columns in a table that uniquely identifies each row.
There is only 1 PK and it should not be NULL.

2. Foreign Key - It is a column / set of columns in a table that refers to the primary key in another table.
There can be multiple FKs.
FKs can have duplicate and NULL values.

- Constraints - They are used to specify rules for data in a table.

__/__/____

i) NOT NULL - column cannot have a NULL value
ii) UNIQUE - all values in column are unique
iii) PRIMARY KEY - makes a column unique and not NULL but used only for one
→ Combination of 2 columns can also be made as a PK.
→ 2 ways of adding PK using CREATE:
i) CREATE TABLE table_name (col1 datatype PRIMARY KEY, col2 datatype constraint, ---);
ii) CREATE TABLE table_name (col1 datatype constraint, col2 datatype constraint, ---, PRIMARY KEY(col1));

iv) FOREIGN KEY - prevent actions that would destroy links between tables
→ CREATE TABLE table1_name (col1 datatype constraint, ---, FOREIGN KEY (col1) references table2_name (col x));

v) DEFAULT - sets the default value of a column.
vi) CHECK - it can limit the values allowed in a column.
→ CREATE TABLE table_name (col1 datatype constraint, ---, CONSTRAINT name CHECK (condition));
→ CREATE TABLE table_name (col1 datatype CHECK (condition));

• Queries -

x) Select some columns of table
→ SELECT col1, col2 FROM table_name;

__/__/____

xi) View distinct records in a table

→ SELECT DISTINCT col_name FROM table_name;

xii) Retrieve data with some specified conditions using WHERE clause

→ SELECT col_name FROM table_name WHERE condition;

⇒ Operators in WHERE clause:

1. Arithmetic Operators - +(addition), -(subtraction), *(multiplication), //(division), %(modulus)

2. Comparison Operators - =(equal to), !=(not equal to), >, >=, <, <=

3. Logical Operators - AND, OR, NOT, IN, BETWEEN, ALL, LIKE, ANY

4. Bitwise Operators - & (Bitwise AND), | (Bitwise OR)

⇒ AND - to check for both conditions to be true

⇒ OR - to check for one of the conditions to be true.

→ WHERE condition1 AND|OR condition2

⇒ BETWEEN - selects for a given range

→ WHERE col_name BETWEEN val1 AND val2;
val1 and val2 are inclusive

__/__/____

⇒ IN - matches any value in the list
→ WHERE col_name IN (val1, val2, ---);

⇒ NOT - negates the given condition
WHERE col_name NOT IN (val1, val2, ---);

xiii) Setting up an upper limit on no. of tuples/rows to be returned using LIMIT clause
→ SELECT col1_name, col2_name FROM table_name
LIMIT number;

xiv) Retrieve data in ascending order using ORDER BY clause
→ SELECT col1_name, col2_name FROM table_name
ORDER BY col_name ASC;

xv) Retrieve data in descending order using ORDER BY clause
→ SELECT col1_name, col2_name FROM table_name
ORDER BY col_name DESC;

- Aggregate Functions - They perform a calculation on a set of values and return a single value.
.. COUNT() - SELECT COUNT(col_name) FROM table_name;
.. MAX() - SELECT MAX(col_name) FROM table_name;
.. MIN() - SELECT MIN(col_name) FROM table_name;
.. SUM() - SELECT SUM(col_name) FROM table_name;
.. AVG() - SELECT AVG(col_name) FROM table_name;

__/__/____

- Group By Clause - It groups rows that have the same values into summary rows. It collects data from multiple records and groups the result by one or more column.

→ It is generally used with some aggregation function

- Having Clause - Similar to WHERE clause i.e. applies some condition on rows. It is used to apply any condition after grouping.

- General Order / Sequence -

SELECT col_name FROM table_name WHERE condition GROUP BY col_name(s) HAVING condition ORDER BY column(s) ASC/DESC ;

- Queries -

xvi) Update existing rows

→ UPDATE table_name SET col_name = val_name WHERE condition ;

* Safe Mode in MySQL - To prevent large amount of changes in database, safe mode gets turned on.

→ UPDATE command needs to be run after :-

SET SQL_SAFE_UPDATES = 0 ;

→ To turn on safe mode again, SET SQL_SAFE_UPDATES = 1 ;

xvii) Delete existing rows

→ DELETE FROM table_name WHERE condition ;

xviii) Delete all rows of a table
→ DELETE FROM table_name;

- Cascading for Foreign Keys -

i) ON DELETE CASCADE - When we create a FK using this option, it deletes the referencing rows in the child table when the referenced row is deleted in the parent table which is PK.

ii) ON UPDATE CASCADE - When we create a FK using this option, the referencing rows are updated in the child table when the referenced row is updated in the parent table which has a PK.

→ CREATE TABLE table1_name (col1_name datatype constraint, col2_name datatype constraint, ---- FOREIGN KEY (col_name) REFERENCES table2_name (col_name) ON DELETE CASCADE ON UPDATE CASCADE;

- Queries -

xix) Add column in a table
→ ALTER TABLE table_name ADD COLUMN col_name datatype constraint;

xx) Remove column from a table
→ ALTER TABLE table_name DROP COLUMN col_name;

xxi) Rename table
→ ALTER TABLE table_name RENAME TO new_table_name;

__/__/____

xxii) Change/Rename column of a table
→ ALTER TABLE table_name CHANGE COLUMN old_name new_name new_datatype new_constraint;

xxiii) Modify datatype/constraint of a column of a table
→ ALTER TABLE table_name MODIFY col_name new_datatype new_constraint;

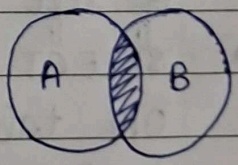
xxiv) Delete table's data using TRUNCATE
→ TRUNCATE TABLE table_name;

- Joins in SQL - Join is used to combine rows from 2 or more tables, based on a related column between them.

⇒ Types of Joins -
1. Inner Join
2. Outer Joins :- i) Left Join
   ii) Right Join
   iii) Full Join

1. Inner Join - Returns records that have matching values in both values.

→ SELECT column(s) FROM tableA
INNER JOIN tableB ON tableA.col = tableB.col;

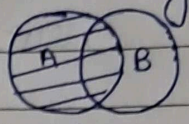



* We can also use alias/alternate name when there are big table names.

E.g: SELECT column(s) FROM tableA AS A INNER JOIN tableB AS B ON A.col = B.col;

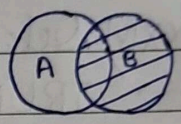
__/__/____

2. LEFT JOIN - Returns all records from the left table and the matched records from the right table.



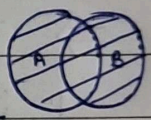

→ SELECT column(s) FROM tableA LEFT JOIN tableB ON tableA.col_name = tableB.col_name;

3. RIGHT JOIN - Returns all records from the right table and the matched records from the left table.




→ SELECT column(s) FROM tableA RIGHT JOIN tableB ON tableA.col_name = tableB.col_name;

4. FULL/FULL OUTER JOIN - Returns all records when there is a match in either left or right table.

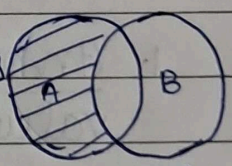



→ SELECT column(s) FROM tableA LEFT JOIN tableB ON tableA.col_name = tableB.col_name

UNION

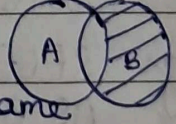
SELECT column(s) FROM tableA RIGHT JOIN tableB ON tableA.col_name = tableB.col_name;

5. LEFT EXCLUSIVE JOIN - Returns records of left table exclusive of right table




→ SELECT column(s) FROM tableA LEFT JOIN tableB ON tableA.col_name = tableB.col_name WHERE tableB.column_name IS NULL;

6. RIGHT EXCLUSIVE JOIN - Returns records of right table exclusive of left table.




→ SELECT column(s) FROM tableA RIGHT JOIN tableB ON tableA.col_name = tableB.col_name WHERE tableA.column_name IS NULL;

__/__/____

7. SELF JOIN - It is a regular join but the table is joined with itself.

→ SELECT column(s) FROM table AS A JOIN table AS B ON A.col_name = B.col_name;

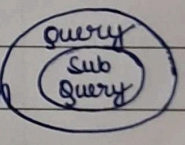
- Union - It is used to combine the result-set of 2 or more SELECT statements. It gives UNIQUE records.

→ To use it : - every SELECT should have same no. of columns
- columns must have similar datatypes
- columns in every SELECT should be in same order

→ SELECT column(s) FROM tableA
UNION
SELECT column(s) FROM tableB;

→ UNION - Removes duplicates
UNION ALL - Doesn't remove duplicate data.

- SQL Sub-Queries - A subquery or nested query is a query within another SQL query. It involves 2 select statements.
Subqueries can be written with SELECT, FROM and WHERE.
↳ Preferred




→ SELECT column(s) FROM table_name WHERE col_name OPERATOR (subquery);

- MySQL Views - A virtual table based on the result-set of an SQL statement is called a view.
table → real → real data → operations
views → virtual

__/__/____

→ A view always shows up-to-date data. The database engine recreates the view, everytime a user queries it.

→ CREATE VIEW view1 AS SELECT column(s) FROM table_name;

SELECT * FROM view1;