SQL

Database:

Database is a collection of data in a format that can be easily accessed.
A software application used to manage DB is called as DBMS.

Types of Databases:

Relational (SQL)
- Data is stored in tables
- MySQL, Oracle, PostgreSQL

Non-Relational. (No-SQL)
- Data stored in excel sheets.
- MongoDB.

What is SQL? - Structured Query Language.

SQL is a programming language used to interact with relational database.
It is used to perform CRUD operations.

Creating our first Database:

```
CREATE DATABASE db_name;      <-- creates a DB
DELETE DATABASE db_name;      <-- Deletes a DB.

USE db_name;                  --> working on this current DB.
```

*Creating our first Table!

```
CREATE TABLE table_name (
    col1 datatype constraint,
    col2 datatype constraint,
    col3 datatype constraint
);
```

*SQL Datatypes:

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

Datatype	Range
CHAR	0 - 255
VARCHAR	0 - 255
BLOB	0 - 65535
INT	-2147483648 to 2147483647
TINY INT	-128 to 127
BIG INT	
BIT	Bit values from 1 to 64.
FLOAT	Decimal value with precision upto 23 digits.
DOUBLE	Decimal number with 24 to 53 digits.
BOOLEAN	0 or 1
DATE	yyyy-MM-DD - 1000-01-01 to 9999-12-31
YEAR	4 digits.

Signed and Unsigned:

By default all INT, FLOAT, DOUBLE are signed.

TINYINT (-128 to 127)

TINYINT UNSIGNED (0 to 255)

Types of SQL commands:

DDL : Create, alter, Drop, truncate.

DQL : Select

DML : insert, update, delete.

DCL : grant & revoke

TCL : commit, rollback, start transaction.

Database related Queries:

CREATE DATABASE db_name;

CREATE DATABASE IF NOT EXISTS db_name;
→ This will prevent from recreating same DB.

DROP DATABASE db_name;

DROP DATABASE IF EXISTS db_name;
→ If present the delete.

SHOW DATABASES; → shows names of existing DB's.

SHOW TABLES; → Shows tables in current DB.

* Table Related Queries:

1. CREATE

```
CREATE TABLE TABLE_NAME(
        column_name datatype constraint,
);
```

2. SELECT and View all columns.

```
SELECT * FROM table_name;
```

↳ * = ALL content.

3. INSERT

```
INSERT INTO table_name (column1, column2)
VALUES (column1_val1, column2_value1),
       (column1_value2, column2_value2);
```

* Keys:

1. Primary Key: It is a column or set of columns that uniquely identifies each row.
There is only 1 Primary Key and it should not be NULL

2. Foreign Key: It is a column in a table that refers to the primary key in other table
There can be multiple FK's. FK's can have duplicate and NULL values.

Constraints :

Used to specify rules for data in table.

1. NOT NULL : column can not have NULL values.

```
col1 INT NOT NULL;
```

2. UNIQUE : all values in column must be different

```
col1 INT UNIQUE;
```

3. PRIMARY KEY : makes the column unique and NOT NULL.

(i)
```
id INT PRIMARY KEY;
```

OR

(ii)
```
id INT,
PRIMARY KEY (id);
```

4. FOREIGN KEY : prevents actions that would destory links between table.

```
CREATE TABLE temp(
    cust_id int,
    FOREIGN KEY (cust_id) references
        customer(id)
);
```

↳ PK in other table.

5. DEFAULT : sets the default value of the column.

```
salary INT DEFAULT 25000;
```

↳ sets salary = 25000 if not specified.

6. CHECK: It can limit the values allowed in a column.

→
```
CREATE TABLE city (
    id INT PRIMARY KEY,
    city VARCHAR(50),
    age INT,
    CONSTRAINT age-check CHECK (age >= 18),
);
```

* Using of operators in WHERE clause:

1. Airthmetic operators: +, −, *, /, %.
2. Comparision operators: =, !=, >, >=, <, <=
3. Logical operators: AND, OR, NOT, IN, BETWEEN, ALL, LIKE, ANY.
4. BITWISE operators: &, |

* LIMIT clause:

Sets an upper limit on number of rows to be returned.

→
```
SELECT * FROM student LIMIT 3;
```

Returns only top3 results.

* ORDER By clause :

To sort in ascending (ASC) or descending order (DESC)

→ SELECT * FROM student
ORDER BY city ASC ;

* Aggregate functions :

These functions perform a calculation on a set of values and return a single value.

I. COUNT () II. MAX() III. MIN()

IV. SUM() V. AVG() .

→ I. SELECT MAX (marks) FROM student ;

II. SELECT AVG (marks) FROM student ;

* GROUP By clause :

Groups rows that have the same value into the summary rows. It collects data from multiple records and groups the result by one or more column.

* Generally we use group by clause with some aggregate function.

→ count no. of students in each city.

```
select city, count(*) as student-count
from student
group by city ;
```

HAVING Clause:

Similar to where clause, applies some condition on rows. Used when we want to apply any condition after grouping.

→ count no. of students in each city where marks cross 90.

```
SELECT city, count(*)
FROM student
GROUP By city
HAVING marks > 90;
```

General syntax Order:

```
SELECT column(s)
FROM table_name
WHERE condition
GROUP BY column(s)
HAVING condition
ORDER BY column(s) ASC;
```

S
F
W
G
H
O

Table related Queries:

1. UPDATE (to update existing rows):

→
```
UPDATE table_name
SET col1 = value1, col2 = value2
WHERE condition;
```

Example:

```
UPDATE student
SET grade = 'O"
WHERE grade = "A";
```

2. DELETE (to delete existing ~~columns~~ rows.):

→ DELETE FROM table_name
WEHERE condition;

Example: DELETE FROM student
WHERE marks < 33;

3. MODIFY column (modify datatype / constraint):

ALTER table_name
MODIFY col_name new-datatype new-constraint;

4. TRUNCATE (to delete table's data):

TRUNCATE TABLE table_name;

~~of~~

DELETE	DROP	TRUNCATE
↓	↓	↓
To delete rows in table	To delete entire table	To delete entire content of table but table still exists.

5. ADD column:

```
ALTER TABLE table_name
ADD COLUMN column_name;
```

6. DROP column:

```
ALTER TABLE table_name
ADD DROP COLUMN column_name;
```

7. RENAME Table:

```
ALTER TABLE table_name
RENAME TO new_table_name;
```

8. CHANGE column (Rename):

```
ALTER TABLE table_name
CHANGE COLUMN old_name new_name new_datatype
                                new_constraint;
```

* Cascading for foreign key:

1. On DELETE Cascade: when we create a FK using this option, it deletes the referencing rows in the child table when referenced rows are deleted in parent table which has primary key.

2. On UPDATE Cascade: when we create a FK using this option, it updates the referencing rows in the child table when referenced rows are updated in parent table which has primary key.

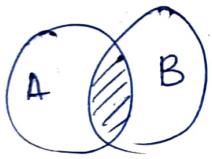
Syntax:

```
CREATE TABLE student (
    id INT PRIMARY KEY,
    courseId INT,
    FOREIGN KEY (courseId) REFERENCES couse(Id)
    ON DELETE CASCADE
    ON UPDATE CASCADE
);
```

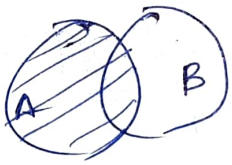
Joins in SQL.

Joins is used to combine rows from two or more tables, based on a related column betwn them.

Types of Joins:

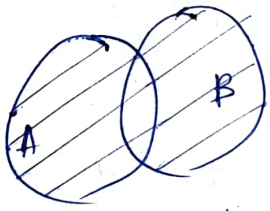


Inner Join



Left Join

Right Join



Full Join

Outer Joins.

UNION:

It is used to combine the result-set of two or more select statements. (Gives unique records).

To use it: i. every select statement should have same no. of columns.

ii. columns must have similar data types.

iii. columns in every select statement should be in same order.

Syntax:

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

SQL Sub Queries:

A subquery or Inner query or a Nested query is a query within another SQL query.

It involves 2 select statement

Syntax:

```
SELECT column(s)
FROM table_name
WHERE col_name operator (subquery);
```

Examples:

Get names of students who scored more than class average.

step 1: Find avg of class.

step 2: Find names of students with marks > avg.

→

```
SELECT name FROM student
WHERE marks > (SELECT AVG(marks) FROM
                student);
```

Example 2: Find the names of all students with even roll no's.

Step1: Find even roll no's.

Step2: Find names of student with even rollno's.

```
SELECT name FROM student
WHERE rollno IN ( SELECT rollno FROM
                  student where rollno%2 =0 );
```

Example 3: Find the max marks of students from Delhi.

Step1: Find students from delhi

Step2: Find max marks.

```
SELECT MAX(marks) FROM student
WHERE city in (SELECT city FROM student
               where city = 'Delhi');
```

MySQL Views

A view is a virtual table based on the result set of an SQL statement.

```
CREATE VIEW view1 AS.
SELECT rollno, name FROM student;
```

} creating the view view1

```
SELECT * FROM view1;
```

} Displaying.

view always shows upto date data.

DBMS

First Normal Form:

* — Table should not contain multivalued attribute

Example:

Roll no	Name	Course
1	Sai	C / C++
2	Harsh	Java.
3	Gojo	SQL / DBMS

} Not in 1st NF

So to convert this table into 1st NF

Roll no	Name	Course
1	Sai	C
1	Sai	C++
2	Harsh	Java
3	Gojo	SQL
3	Gojo	DBMS.

Second Normal Form:

- Table must be in 1st NF
- All the non-prime attributes should be fully dependent on candidate key.

customer_Id	store_Id	Location
1	1	Delhi
1	3	Mumbai
2	1	Delhi
3	2	Banglore
4	3	Mumbai

Candidate key: customer_Id store Id

prime attribute: customer-Id store Id.

Non prime attribute: Location.

Now, according 2NF NP attribute must be fully functional dependent on candidate key

But in above table it is visible that NP attribute location is partially dependent only on store id.

So we can form the table in below manner

Customer Id	Store Id
1	1
1	3
2	1
3	2
4	3

StoreId	Location
1	Delhi
2	~~Mumbai~~ Banglore
3	Mumbai

Example 2 : R(ABCDEF)

FD{ C → F, E → A, EC → D, A → B }

candidate key = EC = FADB

EC^+ = ECFADB

∴ Cand Key = { EC } — step 1

Step 2 : Prime attributes = { E, C }

Step 3 : Non-prime attributes = { A, B, D, F }.

(1) LHS should be proper subset of CK and RHS should be non prime attribute.

Since there is partial dependency is above table it is not in 2NF.

Third Normal Form

- Table should be in 2nd NF
- There should be no transitive dependency in the table.

Rollno	State	City
1	Punjab	Mohali
2	Haryana	Ambala
3	Punjab	Mohali
4	Haryana	Ambala
5	Bihar	Patna

FD = { Rollno → State
State → City }.
NPA NPA

CK = {Roll no}.

PA = Rollno

NPA = State, city.

For each functional dependency,
LHS must be a candidate key and
RHS must be a non-prime attribute.

→ LHS of functional Dependency must be Candidate key / super key OR
RHS must be a prime attribute.

BCNF - Boyce Codd Normal Form.

Rollno	Name	VoterId	age
1	Ravi	K0123	20
2	Varun	M034	21
3	Ravi	K786	23
4	Rahul	D286	21

CK = { rollno, VoterId }

FD = { Rollno → name
Rollno → VoterId
VoterId → Age
VoterId → Rollno }

* Superkey: is a set of attributes of a relation schema upon which all other attributes are functionally dependent.

* SQL commands

DDL	DQL	DML	DCL	TCL
↓	↓	↓	↓	↓
create	select	Insert	Grant	commit
alter		update	revoke	rollback.
~~delete~~		delete		
drop				
truncate.				

* Difference betw HAVING and WHERE clause?

→ I. HAVING is used to specify a condition for a group or an aggregate function used in select statement.

II. WHERE clause selects before grouping.

III. HAVING clause selects rows after grouping.

IV. UNLike, HAVING clause WHERE clause can not contain aggregate functions.

* What is a VIEW?

→ Virtual table based on result set of a SQL statement

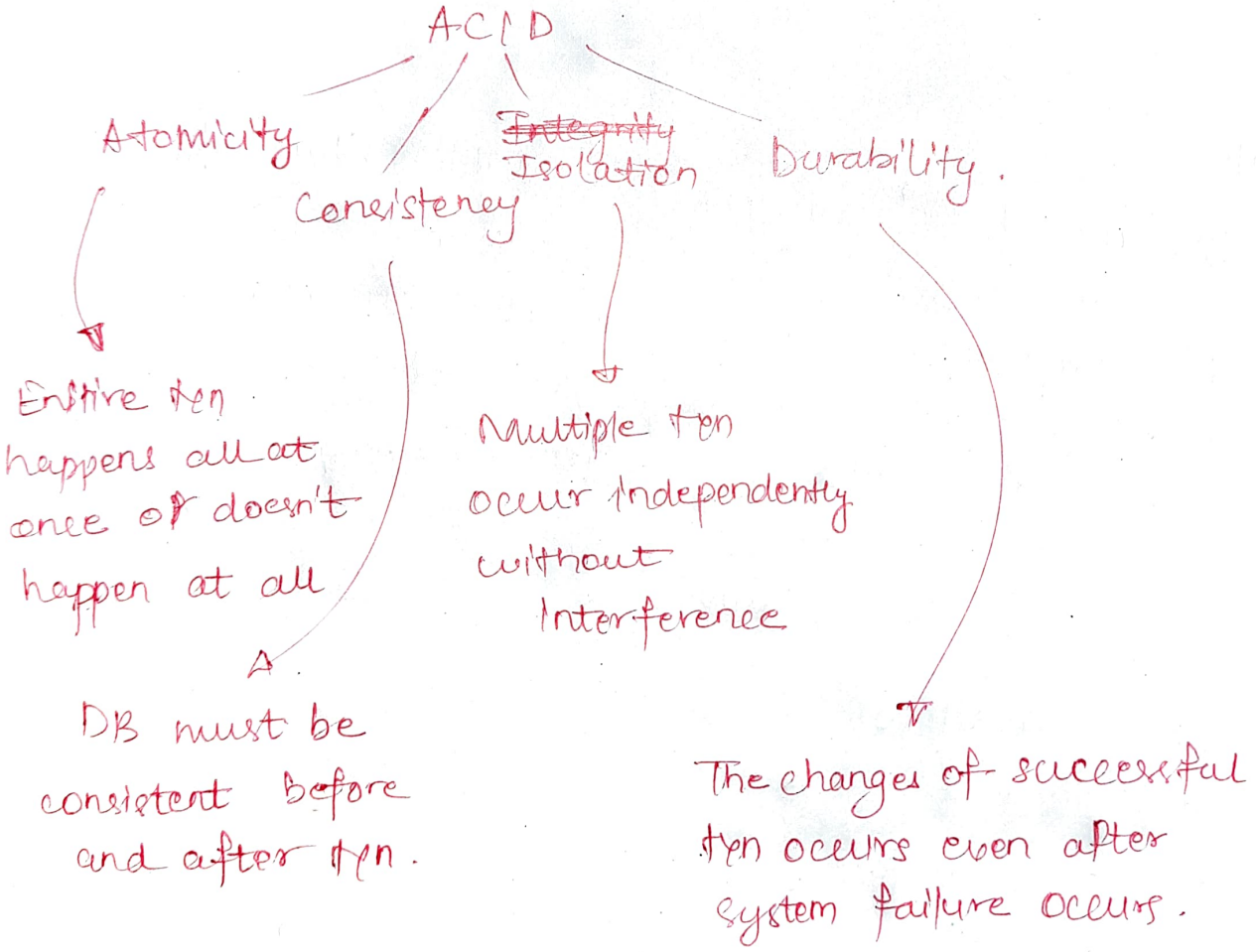
```
CREATE VIEW View1 AS.
SELECT * FROM
Student;
```

* What is a Trigger?

→ A Trigger is a piece of a code associated with insert, update or delete operations.
The code is executed automatically whenever the associated query is executed on table.
Triggers are useful to maintain integrity in the DB.

* What is transaction ? ACID properties ?

→ A DB transaction is an operation / process that must be treated as whole i.e. either all of operation are executed or none of them.



SQL CASE Expressions.

→ Syntax:

```
CASE
    WHEN condition1 THEN result1
    WHEN condition2 THEN result2
    ELSE result
END;
```

Example:

```
SELECT orderId, Quantity,
CASE
    WHEN Quantity > 30 THEN "greater"
    WHEN Quantity = 30 THEN "equal"
    ELSE "less"
END
FROM orderDetails;
```

Rank() function:

assigns position or rank to each row

```
SELECT name, score
RANK() OVER (ORDER BY score) AS RANK
FROM students;
```

- DENSE_RANK():

Similar to RANK() function assigns ranks to rows. But it ensures no gaps between ranks of rows having same value.

- Case Manipulation:

I. LOWER(): Returns string in LowerCase.

II. UPPER(): Returns String in UpperCase.

III. INITCAP(): Returns String in first letter UP and rest in lowercase

- Group function → aggregate functions,

- COALESEC(): If all the expressions evaluate to NULL it returns NULL.

- ISNULL(): used to replace NULL values.

- IFNULL(): takes 2 arguments