⇒ It is standard database language used for accessing & manipulating data in database.

SQL (Structured Query language)

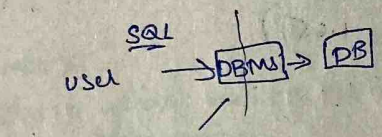
DataBase

↓ digital format

⇒ DataBase is collection of data in a format that can be easily accessed (it is stored in our laptops and system (digital)) format

⇒ A software application we used to manage our DB is called DBMS (Database Management system).

⇒ It is not a programming language, rather it is communicating language



⇒ The SQL we use to interact with DBMS to get the data from the DataBase.

⇒ By using executing queries we can create, update, delete and retrieve data.

DataBases are of two types

① Relational database
② Non-Relational database

Relational → It is also called (Relational DataBase Management System) RDBMS

The data will be stored in tables

Ex: In old days attendence's will would was taken in Register where one column for roll no, one for name, one for present(or)absent.

★ Examples of Relational DataBase Management Systems are :-

① MySQL
② Microsoft SQL Server
③ ORACLE SQL
④ PostgreSQL.

★ Database workbench can be used to view, create and edit tables, indexes, stores procedures and other database meta data object.

★★★ Non-Relational DataBases (NoSQL)

⇒ data that will not be stored in Table format

⇒ mongoDB

SQL : Structured Query language

⇒ It is used to interact with databases

⇒ SQL is a programming language used to interact with relational databases.

⇒ It is used to perform CRUD operations:

① Create — create Database
② Read — To read which is stored in Database
③ Update — It is used update the data in database
④ Delete — It is used to delete the data for from the database

★ SEQUEL — first Name of SQL

Structured English Query Language

Created by IBM Company in 1970s

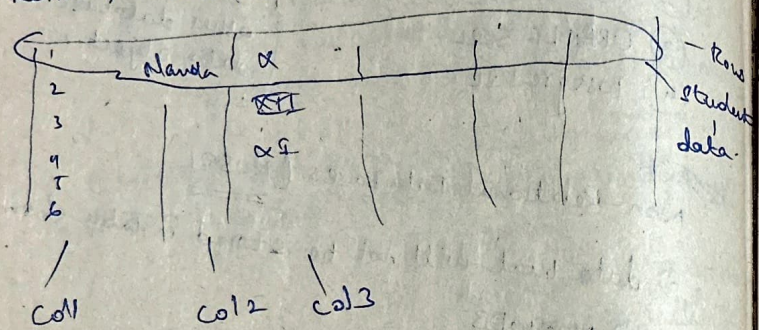
SQL

Structured Query Language

What is a Table?

Student table

Rollno | Name | class | DOB | Gender | City | Marks



Creating our first Database:-

our first SQL Query

1. CREATE DATABASE db_name; — To create database; — Semicolon
2. DROP DATABASE db_name; — To delete the database

⇒ we can write characters in small case also

⇒ SQL is not a case sensitive language.

⇒ But we generally prefer to write them in uppercase letters only so that the letters would be highlighted.

Query

3. USE db_name;

⇒ It used make changes in the particular database.

Creating our first Table:-

USE db_name;

```
CREATE TABLE table_name1(
    Column_name1 datatype constraint,
    Column_name2 datatype constraint,
    Column_name3 datatype constraint
);
```

```
CREATE TABLE Student(
    id INT PRIMARY KEY,
    name VARCHAR(50),
    age INT NOT NULL
);
```

Table Name: Student

id	Name	age

3 columns with different Name

operator	Condition	SQL Example
=, !=, <, <=, >, >=		col_name = 4
Between-AND --		col_name BETWEEN 1.5 AND 10.5
NOT Between		col_name NOT BETWEEN 1 AND 10
IN		col_name IN(2,4,6)
NOT IN(..)		col_name NOT IN (4,5)

```
WHERE birthday Between '1980-05-01'
AND '1980-05-31';
```

AAA

SQL Datatypes

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

DATATYPES	DESCRIPTION	USAGE
CHAR	string(0-255), can store characters of fixed length	CHAR(50)
VARCHAR	string(0-255), can store characters up to given length	VARCHAR(50)
BLOB	string(0-65535), can store binary large object	BLOB(1000)
INT	integer(-2,147,483,648 to 2,147,483,648)	INT
TINYINT	integer(-128 to 127)	TINYINT
BIGINT	integer(-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807)	BIGINT
BIT	can store x-bit Values. x can range from 1 to 64.	BIT(2)
FLOAT	Decimal number - with precision to 23 digits	FLOAT
DOUBLE	Decimal number - with 24 to 53 digits	DOUBLE
BOOLEAN	Boolean Values 0 to 1	BOOLEAN
DATE	date in format of YYYY-MM-DD ranging from 1000-01-01 to 9999-12-31	DATE
YEAR	year in 4 digits format ranging from 1901 to 2155	YEAR

10 digit no

19 digit no

CHAR

⇒ char let say suppose you assign a length of 100 characters can be stored in it

⇒ The Datatype char will block all the place and it can't be reused again.

Eg:-

Mumbai.

string length is only '6'

it will store in 100 and 94 will be blocked because included that you will use it.

VARCHAR

⇒ Varchar it takes only place only based on the length of the char string length (or) word length

Eg:-

Mumbai

M	U	M	B	A	I

BIT(1) → 0,1

BIT(2) → 00 10 01 11

used to store decimal Values.

Signed & Unsigned

TINYINT UNSIGNED(0 to 255)

TINYINT(-128 to 127)

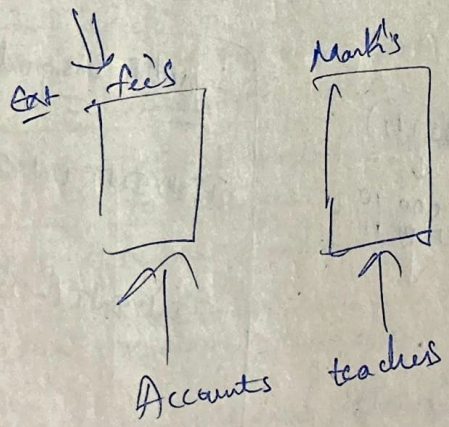
Types of SQL commands :-

① DDL (Data Definition language) :- create, alter, rename, truncate and drop.
- allow to execution of queries like CREATE, DROP, ALTER
- Creation, deletion, addition of Database

② DQL (Data Query Language) : Select

③ DML (Data Manipulation Language) : insert, update & delete
- It is used access (or) manipulate data in the database
- functions will done
  1. Insert
  2. delete
  3. Retrieve
  4. update
- To insert
- delete and edit in the ROW

④ Data
DCL (Data control language) : grant & revoke permission to users



Providing access to the user for only that particular data which they need to see.

⑤ TCL (Transaction control language) : transaction, commit, rollback etc.

will learn advance SQL

Database related Queries :

① CREATE DATABASE db_name;
CREATE DATABASE IF NOT EXISTS db_name;

Write this if it checks if the database is already created it will not the same data base again.

② DROP DATABASE db_name;
DROP DATABASE IF EXISTS db_name;

It checks if exists only it will implement further.

③ SHOW DATABASES;

④ SHOW TABLES;

★ Table related Queries

① Create → table schema (design) → of columns

② creating database
use data_name;

```
CREATE TABLE table_name(
    Column_name1 datatype constraint,
    column_name2 datatype constraint,..
);
```

Tablename - Student

★ Table related Queries

② select & View ALL column's

→ print all

```
SELECT * FROM table_name;
```

③ Insert INSERT INTO table_name VALUES(col1_val, col2_val)

```
INSERT INTO table_name
(colname1, colname2)
VALUES
(col1_v1, col2_v1),
(col1_v2, col2_v2)
```

Multiple ↑ Values

col1 - Values (complete)
col2 - Values

Ex:-

```
INSERT INTO
student
(rollno, name)
VALUES
(101, "kalam"),
(102, "arjun");
```

(or)

```
INSERT INTO Table_name
(col1Name, col2Name, col3Name)
VALUES
(1, "shiva", 20000),
(2, "varun", 30000),
(3, "mahesh", 40000);
```

(or) (1, 2, ..), ("shiva", "varun", "mahesh"), (20000, 30000, 40000)

★ To select and view all your table data

→ SELECT * FROM table_name;

Keys

★ Primary key :- → unique, not null

It is a column (or set of columns) in a table that uniquely identifies each row. (a unique id)
There is only 1 PK and it should not be NOT NULL

Ex:- Take a Table as example in which there are two column's

It should be primary key and NOT NULL

Roll No	Name

Foreign Key:-

A Foreign key is a column in a (or set of columns) in a table that refers to the primary key
There can be multiple FKs.
FKs can be duplicate & null values.

Constraints

⇒ SQL constraints are used to specify rules for data in a table.

(1) NOT NULL - Column's cannot have a null Value (col1 int NOT NULL)

(2) UNIQUE - all Values in a column are different (col2 int UNIQUE)

(3) PRIMARY KEY - make a column unique and not null but used only for one

id int PRIMARY KEY.

id → Column Name; int → Variable name

```
CREATE TABLE temp(
id int not null ;
PRIMARY KEY (id)
);
```

Ex: PRIMARY KEY (id, name)

id	name
101	shyam
101	Ram
102	Ram
103	

If you want to write primary and of two at a time and want to duplicate and name, rollno can't be same in two row different rows.

(4) FOREIGN KEY :- prevent actions that would destroy links between tables.

(It is used to link two different types)

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

cust_id → Column name created in new table

customer (id) → Taking reference

The table which you are taking as reference it should be primary key in table which you are Refering.

(5) DEFAULT set the default Value of a column

Salary INT DEFAULT 25000

Salary → Column Name; INT → Variable Name; DEFAULT → Type

(6) CHECK it can limit the Values allowed in a Column

(1) CREATE TABLE city(
id INT PRIMARY KEY,
city VARCHAR(50),
age INT,
CONSTRAINT age_check (checks the condition)
CHECK (age>=18 AND city="Delhi")
);

checks whether age is above or not if not data will not stored in the database

CREATE TABLE newTab(
age INT CHECK (age>=18)
);

Select in Detail

used to select any data from the database

Basic syntax

SELECT col1, col2 FROM table_name;

Basically we use select all function

SELECT * FROM table_name;

we use to show the data on the screen.

SELECT DISTINCT colname table_name;

unique and in this some names will not repeated.
(col) duplicates

Where clause

the where statement allows us to specify conditions on columns for the rows to be returned

To define some conditions

SELECT col1, col2 FROM table_name
WHERE conditions;

Ex:
SELECT * FROM student WHERE marks > 80;
SELECT * FROM student WHERE city = "Mumbai"

By using it would become a condition, to check.

SELECT *
FROM table_name
WHERE col > condition AND col = condition;

It used to join the condition and find whether it is true or not

Addition
Ex:
SELECT column_name+100 FROM table_name

Using operators in WHERE

Arithmetic operators : + (addition), - (subtraction), * (multiplication), / (division), % (modulus)

comparison operators : = (equals to), != (not equal to), >, >=, <, <=

Logical Operators : AND, OR, NOT, IN, BETWEEN, ALL, LIKE, ANY

Bitwise Operators : & (Bitwise AND), | (Bitwise OR)

Operators:

AND SELECT * FROM table_name WHERE condition1 AND condition2
OR SELECT * FROM table_name condition1 OR condition2

BETWEEN (selects for a given range)
SELECT * FROM table_name WHERE col BETWEEN cond1 to cond2;

→ IN (matches any Value in the list)

SELECT * FROM table_name WHERE IN (city, col name)
(("DELHI", "MUMBAI"))

It checks if the above conditions are satisfied then it will fetch the data

→ It gives the data the cities which we mentioned

→ NOT (to negate the given condition)

SELECT * FROM table_name WHERE city NOT IN ("Delhi", "MUMBAI");

It will gives the values of the data which is not mentioned in the it.

→ LIMIT clause:

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

(i) SELECT * FROM table_name LIMIT 3;

(ii) SELECT col1, col2 FROM table_name LIMIT number;

SELECT col1, col2 FROM table_name LIMIT number;

Order By clause:-

To Sort ascending (Asc) (or) descending order (DESC)

SELECT * FROM table_name ORDER BY city Asc; (1 2 3 4 5 → Ascending)
(Ascending)

SELECT * FROM table_name ORDER BY city Asc; → Ascending
(column name)

(5 4 3 2 1 → Descending)

SELECT * FROM table_name ORDER BY column_name DESC → Descending

→ Aggregate functions:

Aggregate functions perform a Calculation on a set of Values, and return a single Value

(1) COUNT()
(2) MAX()
(3) MIN()
(4) SUM()
(5) AVG()

Get Average marks

SELECT avg (marks) FROM student;

Ex:

Get Maximum marks

SELECT max(marks) FROM student;
(function name: max; column name: marks; table name: student)

Group By clause

Groups rows that have the same values into summary rows.

It collects data from multiple record records and groups the result by one or more column.

⇒ Generally we use group by with some aggregation function.

Ex:-

count number of students in each city

```
SELECT city, count(name)
       column name   Aggregation
FROM student
     Table_name
GROUP BY city;
         column name
```

Q. Write the query, to find avg marks in each city in ascending order.

```
SELECT city, avg(marks)
       column name
FROM student
     Table_name
GROUP BY city
ORDER BY city;
```

Ascending order

Having clause

Similar to where i.e applies some condition on rows.

Used when we want to apply any condition after grouping. ⇒ checking condition.

⇒ "where" we use in Row only

Basically

Example of Having clause :-

```
SELECT count(name), city
       column_name  column_name
FROM student
     Aggregation function
GROUP BY city
HAVING max(marks) > 90;
       column_name
```

checking how many student marks crossed 90.

General order :-

```
SELECT column(s)
FROM tablename
WHERE condition  — Row condition
GROUP BY column(s)
HAVING condition  — Aggregation's condition
ORDER BY column(s) ASC;
```

Check conditions

groups

Table related Queries:-

UPDATE

Update (to update existing rows)

```
UPDATE table_name
SET col1 = val1, col2 = val2
WHERE condition;
```

Example to change the grade to "O" from "A"

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

while you try to update it shows a error of

Error 1175 code

```
SET SQL_SAFE_UPDATES = 0;
```

off (safemode)

```
SET SQL_SAFE_UPDATES = 1;
```

DELETE

DELETE (to delete existing rows)

```
DELETE FROM table_name
WHERE condition;
```

Ex:-

```
DELETE FROM table
WHERE marks < 33;
```

⇒ Should be careful while we need to be using Delete operation once the operation is performed and the data can't be retrieved back.

FOREIGN KEY

FOR

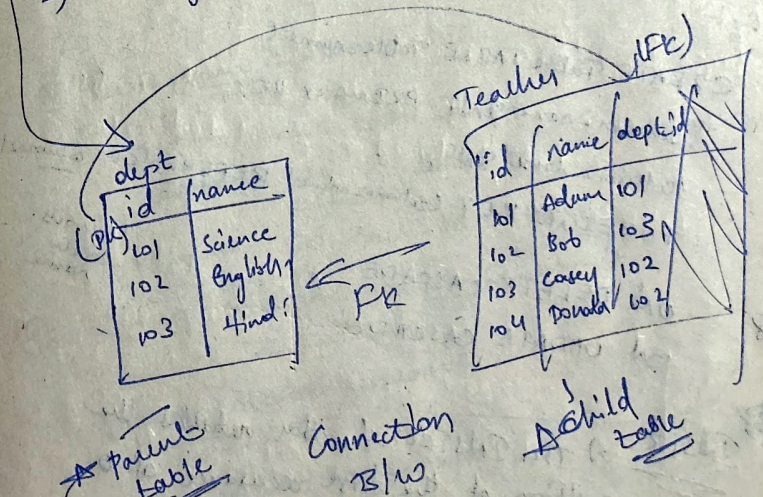
foreign Key

```
CREATE TABLE table_name (
column1_name PRIMARY KEY,
column2_name datatype constraint,
column3_name datatype constraint,
FOREIGN KEY (column name) REFERENCES
Table name (Previous Column Name)
);
```

Previous

⇒ We can Visualise it through SQL WorkBench settings in that database

⇒ Using ER diagrams



Cascading

Cascading for FK :-

on DELETE cascade :-

When we create a foreign key using this option, it deletes the referencing rows in the child table when the referenced row is deleted in the parent table which has a primary key.

on update cascade :-

When we create a foreign key using UPDATE CASCADE the referencing rows are updated in the child table when the refrenced row is updated in the parent table which has a primary key

Ex :-

```
CREATE TABLE TABLE table_name(
    column1_name INT PRIMARY KEY,
    column2_name INT,
    FOREIGN KEY (column2_name) REFERENCE column2(id)
                                        ↑
                                    Previous table name
    ON DELETE CASCADE
    on UPDATE CASCADE
```

INSTR

1) The INSTR() function returns the position of the first occurance of a string in another string

Syntax

```
SELECT INSTR(column_name, 'a')
FROM table
WHERE column_name = "shivanki";
```

Table related Queries

Alter (to change the schema)

↳ design (column's, datatype, constraints)

To Bring changes in the table design

(1) TO ADD column

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

(2) TO Drop column

~~DROP TABLE table_name~~

OR

```
ALTER TABLE table_name
DELETE COLUMN column_name datatype (constraint
```

(3) TO Rename ~~column~~ Table

```
ALTER TABLE table_name
RENAME COLUMN column_name datatype constraint
RENAME TO new_table_name
```

(4) To change column (name) (Rename)

change column (rename)

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

⑤ Modify column (modify datatype/constraint)

```
ALTER TABLE table_name
MODIFY col_name new_datatype new_constraint;
```

Truncate (to delete table's data)

```
TRUNCATE TABLE table_name;
```

→ It is used to delete the data from table but it will not delete ~~data from the table~~

→ Table remains with empty data

Joins in SQL :-

→ Joins is used to combine rows from two or more tables, based on a related column's between them.

Eg →

Employee

id	name
101	—
102	—

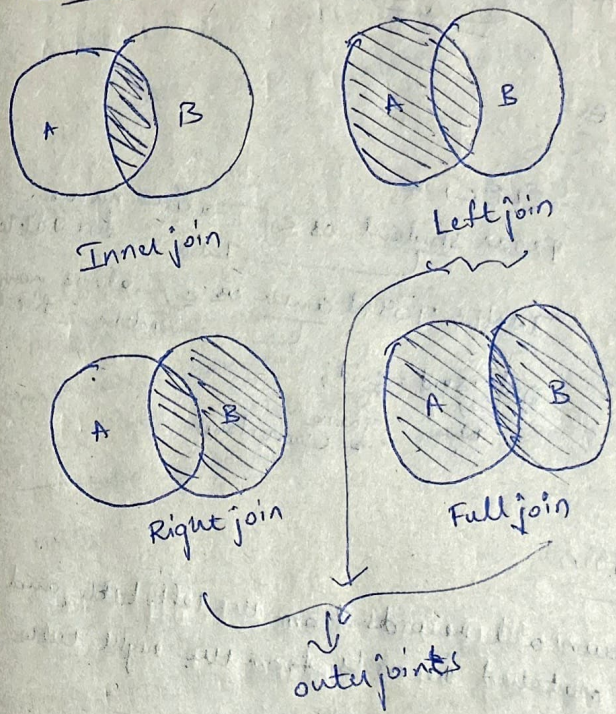
Salary

id	Salary
102	
103	

Taking data of same :

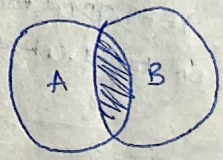
It's not mandatory to join it using foreign key.

Types of joins :- (Venn diagram)



Inner join

Return's records that have matching values in both tables



★ Syntax

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

Table A ⇔ Table B

Alias

↳ Alternable name

Ex:-

```
SELECT *
FROM student as s      (alias name for Table1)
INNER JOIN course as c (alias name for Table 2)
ON s.id = c.id;
```

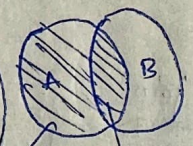
s → Table 1, c → Table 2

Left Join

Return all records from the left table, and the matched records from the right table.

Syntax:-

```
SELECT column(s)/or1 *
FROM table1_name
LEFT JOIN table2_name
ON table1.col_name = table2.col_name;
```



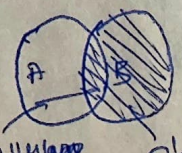
left full And overlapping data

Right Join :-

Return all matched records from the left table and all records of Right table

Syntax:-

```
SELECT column(s)/or1 *
FROM table1_name
RIGHT JOIN table2_name
ON table1.col_name = table2.column_name;
```

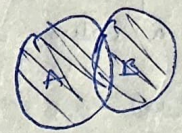


Overlapp data — Right Join

Full Join

Returns all records when there is a match in either left or right table

Syntax in MySQL

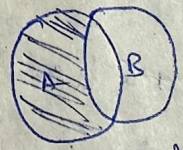


Left Join UNION Right Join

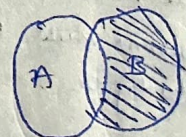
```
SELECT *
FROM table1_name as any_char
ON char_column_name =
LEFT JOIN table2_name as any_char
on table1_name.char1 = table2_name.char2
UNION
SELECT *
FROM table1_name as any_char
RIGHT JOIN table2_name as any_char
ON table1_name.char1 = table2_name.char2;
```

---

② Write SQL commands to display the right exclusive join :



Left Exclusive join



Right Exclusive join

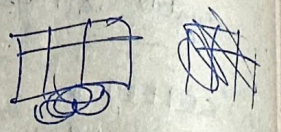
Syntax:-

```
SELECT *
FROM student as a
LEFT JOIN course as b
ON a.id = b.id
WHERE b.id IS NULL;
```

```
SELECT *
FROM student as a
RIGHT JOIN course as b
ON a.id = b.id
WHERE a.id IS NULL
);
```

Self Join

- A self join is a regular join in which a table is joined to itself.
- Self joins are powerful for comparing values in a column of rows with the same table.

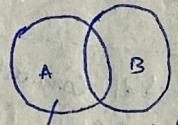


Syntax:

```
SELECT column_name(s)
FROM Table AS T1
JOIN Table AS T2
ON T1.col_name = T2.col_name
```

UNION

It is used to combine the result-set of two or more SELECT statements.
Gives UNIQUE records



tableA <-> tableB

To use it:

- every SELECT should have same no. of columns
- columns must have similar data types
- Columns in every SELECT should be in same order

Syntax:- UNION → It will not print duplicates

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

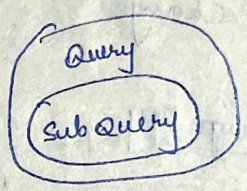
Syntax: UNION ALL

```
SELECT column(s) FROM tableA
UNION ALL
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 statements.



Syntax:-

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

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;
SELECT * FROM view1;
```

(View1 — Virtual table)

table → real tables → real data → operations → database

Example take a Bank the person who works in sales wi should acess every data to be accessed by him just gretal general details like name, contact info and data of account activation and for them we can create Virtual and limited database

Views → Virtual

Primary key — IP Points

⇒ The primary key is a column or a set of column that uniquely identifies each row in the table.

⇒ A table can have only primary key which can be made up of one or more fields.

⇒ The primary key must contain unique values should not be null and should never change.

Foreign key :-

⇒ Foreign key is a column or set of column's that is used to link two tables together.

⇒ The foreign key of one tables helps to connect with the primary key of another table.

⇒ Foreign key is used to create links b/w tables.

⇒ It is used to establish a link b/w two tables.

⇒ Sy CREATE TABLE student (
id INT PRIMARY KEY,
name VARCHAR,
Salary INT NOT NULL
);

CREATE TABLE Studentdata (
parentName VARCHAR(),
id INT,
FOREIGN KEY(id) REFERENCES student(id)
);

⇒ FORF
foreign key in one table used to point primary key in another table.

⇒ A table has a combination of rows and column's. Rows are called records and column's are called fields.

⇒ In MYSQL Server the tables are being designated within the database and schema names.

Primary key :-

It is also called primary keyword, is a column in a relational database

⇒ Primary keys serve as unique identifiers for each row in a database table.

Default :-

⇒ Default constraint is used to fill a column with default and fixed values.

(2)

SQL queries are powerful tools used to manipulate and manage data stored in these databases like mysql, oracle, postgresql

(2) Query

Query → To print in upper case letters from and table and considering as ALIAS with another name.

Answer

upper case indication

```
SELECT upper(FIRST_NAME) as
STUDENT_NAME from student;
```

STUDENT_NAME → ALIAS

upper(FIRST_NAME) → Existing Name will be converting to Alias

as → will be used to convert alias

(2) SQL Query for to print first 3 characters of a column from table

Syntax

```
SELECT SUBSTRING(column_name,1,3)
FROM table;
```

(2) Delete

SELECT

INSTR() function returns the position of the first occurance of a string in another string.

Ex

```
SELECT INSTR(FIRST_NAME,'a')
FROM student
WHERE FIRST_NAME = 'shivansh';
```

To print the unique values from a column i.e in that no multiples should be there

Ex

```
SELECT DISTINCT MAJOR, LENGTH(MAJOR) FROM
STUDENT;
```

DISTINCT MAJOR → Table name

LENGTH(MAJOR) → To find the length of the column value.

```
SELECT REPLACE(FIRST_NAME,'a','A')
FROM student;
```

→ It is used to replace the word in the table.

for only one name.

```
SELECT REPLACE(FIRST_NAME,
'a','A')
FROM student
WHERE FIRST_NAME='shivan';
```

operator	condition	Example
=	Checks is it exact or not	col_name = "abc"
!= or <>	It returns the values which are not equal to the given value	col_name != "abcd"
LIKE	Same give exact string	col_name LIKE "ABC"
NOT LIKE	Notes gives the string which you have given!	col_name NOT LIKE "ABC"
_	used to match a single character	"AN_" matches "AND" but not "AN"
IN (..)	string exist in a list	col_name IN ("A", "B", "C")
NOT IN (..)	string does not exist in list	col_name NOT_IN ("D", "E", "F"