Data Manipulation Language (DML):

This language is mainly concentrated towards manipulating data stored inside tables in databases.

Eg: Select, insert, update, delete

→ Inserting value inside a table:

* to insert one row at a time →

insert into table-name (column1, column2, column3 ...)
values (value1, value2, value3, ...);

** to insert multiple rows at a time →

insert into table-name (column1, col2, col3) values
(val1, val2, val3), (val11, val22, val33), (val111, val222, val333);

→ Updating data that has been inserted into table:

** updating a single row -

update table-name set column-name = 'value' where
column-name = 'value';

↳ always should be primary key column.

* To use different column other than the primary key, we need to turn off the safe updates in MySQL that can be done as -

set sql-safe-updates = 0;

** updating multiple columns values of a single row -

update table-name set column-name1 = 'value1', column-name2 = 'value2' where column-name3 = 'value3';

** updating multiple row values -

update table-name set column-name = 'value' where column-name2 = 'value2'

→ Deleting data from the table —

** Deleting one row & multiple rows —

delete from table_name where column-name = 'value';

** Delete every single row from our table:

delete from table_name;

→ Selecting from a table:

* To select all the records in a table, we use
select * from table-name;

* To select a particular column and its all the records, we use
select column-name from table-name;

* To select multiple columns and all their records, we use
select col_name1, col_name2, col_name3, --- from table-name;

* To select multiple rows from a column or entire rows which fulfill a certain criteria, we use 'where' clause.

Eg: select * from table-name where col-name = value';

We can have multiple where clauses in a SQL statement:

Select * from table_name where col-1 = val1 and/or
col2 = val2 and/or col3 = val3 --;

→ We can even use relational operators like <, >, >=, <=, != with the where clause. They are used when dealing with numeric values.

Eg: select * from table-name where col-name < value;

→ Retrieving null records from a column:

select * from table-name where col-name is null;

* Doing just opposite —

select * from table-name where col-name is not null;

Exercise 2

(1) From the customers table, select the first name and phone number of all the females who have last-name of Bluth.

Ans→ Select first-name, phone-number from customers where last-name = 'Bluth' and gender = 'F';

(2) From the products table, select the name for all products that have a price greater than 3.00 or a coffee-origin of Srilanka.

Ans→ Select name from products where price > 3.00 or coffee-origin = 'Srilanka';

(3) How many customers don't have a phone number entered into the customers table —

Ans→ Select * from customers where gender = 'M' and phone number is null;

→ IN, NOT IN Key words

* To select a set of values from a column we can use IN & NOT IN operators.

Select first-name from customers where last-name IN ('Gupta', 'Verma', 'Singh');

Op: Saurabh, Rajneesh, Sachin } Full name → Saurabh Gupta, Rajneesh Singh, Sachin Verma

* Select all the first-name where last-name is not Gupta, Verma, Singh.

select first_name from customers where last-name not in ('Gupta', 'Verma', 'Singh').

→ Between Keyword (Can work with dates, numbers & strings)

To select and return values between a certain range.

select column-name from table-name where column-name between val 1 and val 2;

→ **Like** statement for pattern matching with where clause:
Can be used on both strings and numbers.
Find all the last-names in a table having their initials from S.

select * from table-name where last-name like 'S%';

Find all the product names in product column, such that their price is 3 or more but not less than or greater than 3. {3.25, 3.18, 3.00, --}

Select name from product where price like '3%';

% → is used for any no. of occurence of characters.
_ → is used for only one character.

Find all names of customers in customer column having o in the first name and has only 3 words (o is in middle).

Select name from customer where first-name like '_o_';

→ **Orderby keyword**: (can be done for numbers, strings, date).

select * from table-name **orderby** col-name asc/desc;

Exercise-3

① From the products table, select the name and price of all products with a coffee origin equal to Colombia or Indonesia. Ordered by name A-Z.

select name, price from products where coffee-origin IN ('Colombia', 'Indonesia') orderby name ASC;

② From the orders table select all the orders from February 2017 for customers with id's of 2, 4, 6 or 8.

select orders from orders where order-date between '2017-02-01' and '2017-02-28' and id in (2,4,6,8);

③ From the customer table select first name and phone number of all customers whose last name contains the pattern 'ar'.

Select first-name, phone-no from customer where last name like '%ar%';

→ **Distinct keyword**
To select/retrieve distinct items from a record/column
distinct

```
Select customer-id from orders where order-time between
'2017-02-01' and '2017-02-28';
```

→ **Limit keyword** → To retreive limited no. of rows using the select statement.

Eg: select * from customers **limit** 5;

→ **OFFSET keyword** →
We can also set an offset value to our limit as—
Select * from customers limit 5 offset 5; returns rows by 6th row no.

→ **Column name alias** ⇒ (AS keyword) (only works in result set)
Select name as coffee and coffee-origin as country from table;

Exercise-4

(1) From the customers table select distinct last name and order alphabetically from A-Z.

Ans→ select distinct last-name from customers order by last-name asc;

(2) From the order table, select the first 3 orders placed by customer with id 1, in February 2017.

Ans→
```
select * from order where id=1 and order-time between '2017-02-01'
and '2017-02-28' order by order-time limit 3;
```

(3) From the products table select the name, price and coffee origin but rename the price to result-price in result set.

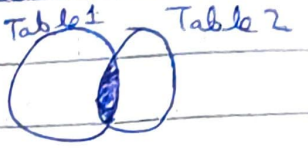
Ans→
```
Select name, price as result-price, coffee-origin from products;
```

Selection from different tables

What is a join?

Ans→ Joins allow you to retrieve data from multiple tables in a single select statement.

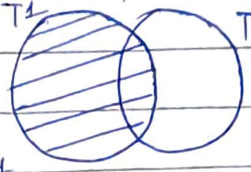
To join two tables there needs to be a related column between them. There are different types of joins that are discussed below – (i) inner join



It retrieves data from both tables if there are matching values in both tables.

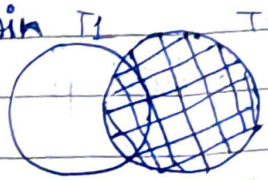
If we want to retrieve data from table 1 and there are null values in table 2, no data will be retrieved and vice-versa

(ii) **left join:**



It retrieves all the data from table 1 and only matching data from table 2.

(iii) **right join**



It retrieves all the data from table 2 and only matching data from table 1.

*MySQL doesn't have full join / outer join.

(i) **Inner join**

Suppose, we have two tables products & orders

products.

id	name	price	Coffee-origin
1	Expresso	2.50	Srilanka
2	Cappuccino	3.50	Africa
3	Latte	3.00	Brazil

orders

id	product-id	customer-id	order-time
2	1	2	—
3	5	3	—
4	3	22	—
1	4	3	—

If we want to select which product was ordered at what order-time, then we need to do joining of these fields in both the tables, as 'id' column in both the table is common, we can use the inner join

```
select products.name, orders.order-time
from product
inner join products on products.id = orders.product-id;
```

Short hand notation:

```
Select p.name o.order_time from orders o join products p on
    o.product = p.id;
```

Customers

id	first_name	l_name	ph_no	gender

→ **left join :** It returns all the data from table 1 and matching data from table 2.
Joining customers.id in orders table and id column in Customers table.

> * Update a value in a record in a row -
> Update orders set customer_id = NULL where id = 1;

select o.id, c.ph_no, c.l_name, o.order_time from orders o left join customers c on o.customer_id = c.id order by o.order_time;

→ **right join :** It returns all the data from table 2 and will return only matching data from table 1.

Select o.id, c.ph_no, c.l_name, o.order_time from customer c right join orders o on c.id = o.customer_id order by o.order_time;

→ **Joining more than two tables**

In our three tables, product_id & customer_id field in orders table is foreign key to products and customers table respe. So, we can join orders table to products table and customers table in same select statement and retrieve data from all tables.

Select p.name, p.price, c.first_name, c.l_name, o.order_time from products p join orders o on p.id = o.product_id join customers c on c.id = o.customer_id.

* products and orders table has common table & customers and orders table had common tables, so we joined accordingly.

Exercise-5

Ques. ① Select the order id and customer phone number for all orders of product id 4.

Select O.order-id, C.ph-no from orders o join customers c on O.customer-id = C.id where O.product-id = 4;

Ques. ② Select productname and order time for filter coffees sold between january 15th 2017 and February 14th 2017.

Select p.name, O.order-time from products p join orders O on p.id = O.product-id where p.name = 'filter' and O.order-time between '2017-01-15' and '2017-02-14';

Ques. ③ Select the product name and price and order time for all orders from females in january 2017.

Select p.name, p.price, O.order-time from products p join orders O on p.id = O.product-id join customers C on O.customer-id = C.id where C.gender = 'F' and O.order-time between '2017-01-01' and '2017-01-31';