**HOW TO CREATE TABLES:**

CREATE TABLE name (

Table1 INT AUTO\_INCREMENT PRIMARY KEY,

Table2 VARCHAR(10) NOT NULL,

Table3 DATE

);

**HOW TO INSERT DATA:**

INSERT INTO name

(Table1, Table2, Table3)

VALUES (1, ‘babo’, ‘2003-05-12);

(You don’t need to write the names of the tables when you insert data for the whole tables)

INSERT INTO name

VALUES (1, ‘babo’, ‘2003-05-12);

(You can insert data for some tables but cannot skip NOT NULL)

INSERT INTO name

(Table1)

VALUES(4, ‘wefw’);

(You can add multiple data as well)

INSERT INTO name

(Table1, Table2, Table3)

VALUES

(4, ‘ad’, ‘1232-23-12);

(5, ‘aef’, ‘1323-23-23);

**THINGS THAT WERE COVERED IN THE CLASS:**

How to use alias: SELECT tablename **AS** whatevernameyouwant

How to sort data: ORDER BY tablename **ASC / DESC** (Default is ASC and you don’t need to write)

String modify: SELECT **CONCAT(firstname, ‘ ’, lastname)** AS name

Show only specific amount of data: **LIMIT**

Table

Description automatically generated

**LIKE:**

Graphical user interface, text, application

Description automatically generated

**JOIN:**

Diagram

Description automatically generated with low confidence

How to use table aliases when join:

FROM vendors **v** JOIN invoices **i**

ON **v**.vendor\_id = **i**.vendor\_id

**INNER JOIN**: Shows only matching data in the both tables.

**NATURAL JOIN**: A NATURAL JOIN returns only the matching rows from both tables based on the columns with the same name in both tables.

NATURAL JOIN does not need an ON clause. This is because it automatically matches the column name. Therefore, if the column names vary, an INNER or OUTER JOIN is preferred.

Diagram

Description automatically generated

**CROSS JOIN**: A CROSS JOIN returns the Cartesian product of the two tables, which includes all possible combinations of rows from both tables.

Diagram, table

Description automatically generated

**LEFT / RIGHT JOIN:** All the data in the LEFT table / RIGHT TABLE

Table

Description automatically generated Table

Description automatically generated

UNION: get all data in 2 tables but in one column (same datatype)

Employ table

|  |  |
| --- | --- |
| Id | name |
|  |  |

Branch table

|  |  |
| --- | --- |
| Id | B\_name |
|  |  |

|  |
| --- |
| Name |
| Name1 |
| Name2 |
| .... |
| B\_name1 |
| B\_name2 |

**Chart, box and whisker chart

Description automatically generated**

**SELF JOIN**: When you are trying to get information from the same table

Table

Description automatically generated

SELECT  
 cust.customer\_id,  
 cust.firstname,  
 cust.lastname,  
 cust.birthdate,  
 cust.spouse\_id,  
 spouse.firstname AS spouse\_firstname,  
 spouse\_lastname AS spouse\_lastname  
FROM **customer** AS cust  
**INNER JOIN** **customer** AS spouse  
   **ON** cust.spouse\_id **=** spouse.customer\_id

Table

Description automatically generated

What’s happening?

You duplicated the customer table and named it ‘spouse’.

You gave alias for the original table(customer > cust), and the copied table(customer > spouse) and inner joined (shows only matching data) them.

And you selected the firstname and the lastname tables from the spouse table to display.

Diagram

Description automatically generated

**Using:**

Text

Description automatically generated

Graphical user interface, application

Description automatically generated

**Aggregate function**

AVG – average

SUM – sum

MIN – minimum value

MAX – maximum value

COUNT – counts rows

**GROUP BY**

Difference between WHERE and HAVING

WHERE: Select specific data from the entire data

HAVING: Additional filtering for the result that was already filtered by GROUP BY

SELECT > FROM > WHERE > GROUP BY > HAVING > ORDER BY

