SQL

--- conditions

greater than >

greater than or equal >=

lower than <

lower than or equal <=

equal =

none of the statement !=

not located in this a place <>

1.---- retrieve the Column from the table

SELECT

---- if you want to sensify the column you want to get

SELECT first\_name , last\_name

2.---- key word to execute the database

USE

3.---- filter to sort data

FROM

4.---- filter the data

WHERE

----- let’s say we only want to get the customer with points > 3000

WHERE point >3000

5.---- we only get one record from the table

ORDER BY

--example (so here you can order by other columns, does not have to be the PK witch the table is always sorted by) here are they sorted by their first name in ascending order.

SELECT\*

FROM customer

ORDER BY first\_name

--- if you want to reverse the sort order, simply type out DESC for descending (synkende)

SELECT\*

FROM customer

ORDER BY first\_name DESC

--example 2

SELECT\*

FROM customer

ORDER BY first\_name DESC, state DESC

6.---- the between operator

BETWEEN

---example

WHERE points BETWEEN 1000 AND 3000

7.----the like operator

LIKE

---example how to get all the name that starts with b

WHERE last\_name LIKE “b%”

----- %b% has b in their name (use the % to present any number of characters)

----- “\_y” where the last name has two characters and ends with y you can also write it “\_\_\_\_\_\_y” or “b\_\_\_\_\_Y” ( \_ to present a single character)

--example you want to get the customer with trail in their address

SELECT \*

FROM Customers

WHERE address LIKE “%trail%”

--example you want to get the customers that their phone number end up with 9

SELECT \*

FROM Customers

WHERE phone LIKE “%9”

8.--- IN operator

SELECT \*

FROM Customers

WHERE state IN (“VA”, “FL”)

9---- logical operator AND, OR, NOT

multiple search conditions, for filtering data

---- has a higher precedence (this will always be executed first, because AND is always valuated first) you can always change the order by using ()

AND

--- example

SELECT\*

FROM customers

WHERE “birth\_date > “1990-01-01” OR points >1000 AND STATE =“VA”

---- is just or

OR

---example (This ex picks up customers who are either born after 1990 or have at least 1000 points)

SELECT\*

FROM customers

WHERE birth\_date > “1990-01-01” OR points >1000

--- to negate a condition (that are not in the table you are searching)

NOT

---example

WHERE state IN (“VA”, “FL”, “GA”)

WHERE state = “VA” , state= “FL” , state=”GA”,

---- or not in ,example

WHERE state NOT IN (“VA”, “FL”, “GA”)

WHERE state = “VA” , state= “FL” , state=”GA”,

10.--- regular expressions, is great when it comes to search for a string. so they allow us to search for more complex patterns

REGEXP

--- example

SELECT \*

FROM customers

WHERE last\_name LIKE “%field%” (this line is identical to the line under)

WHERE last\_name RECEXP “field”

“field$” represents an end of a string (for example the last name must end with field)

“^field” represents the start of a string (must start with field)

“field|mac” finds both field and mac in the last name(we use a multiple bar (the bar means OR) to find more patterns)

“[]” use to match any single characters listed in the brackets

“[A--H]” to represent a rage

Et bilde som inneholder tekst

Automatisk generert beskrivelse

Et bilde som inneholder tekst

Automatisk generert beskrivelse

Et bilde som inneholder tekst

Automatisk generert beskrivelse

11.---- how to get the records with missing values

NULL

--- example

SELECT \*

FROM customers

WHERE phone IS NULL

12.---- DESC descending order (not alphabetic)

synkende rekke

13.---- ASD ascending order

14.---- sequence of character

--the “”

WHERE state = “va”

15----- the limit (can use if you only want to get for example the first 3 customers)

SELECT \*

FROM customers

LIMIT 3

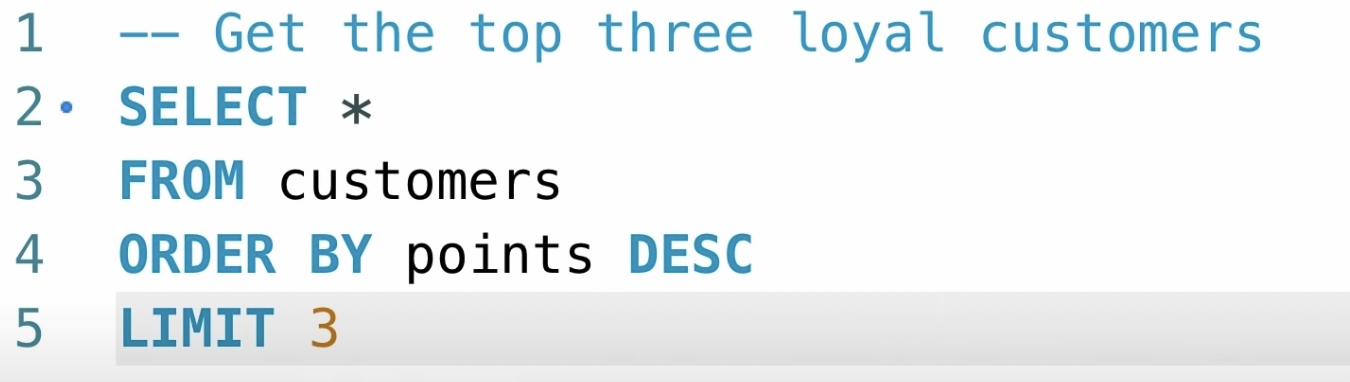
13.---- let say you want to get to customer 7-8-9 in the table

SELECT \*

FROM customers

LIMIT 6, 3

--- you first limit the 6 first (is an offset) basically tells the program to skip the first 6 records. and then pick the 3 first records.



--- the limit clause should always come at the end

INNER JOINS, OUTER JOIN

14----- Inner joins (how to select the orders in the orders table)

SELECT \*

FROM orders

INNER JOIN customers

ON orders.customer\_id = customers.customer\_id

--- after on you must type a condition (betingelse)

---in situation where you have the same column in multiple tables , you need to qualify them by prefixing them with the name of the table , in this example its orders

SELECT order\_id, orders.cutomer\_id, first\_name, last\_name

FROM orders

INNER JOIN customers

ON orders.customer\_id = customers.customer\_id

--- example 3 apply an alias , so when you have the same condition, such as orders you can apply an alias such as o instead of writing the condition

SELECT order\_id, o.cutomer\_id, first\_name, last\_name

FROM orders o

INNER JOIN customers c

ON o.customer\_id = c.customer\_id

14.---- joining across databases (how to combine columns from tables in multiple databases)

USE sql\_inventory;

SELECT \*

FROM order\_items oi

JOIN sql\_invetory.products p

ON oi.product\_id = p.product\_id

---- you only must prefix the tables that are not part of the current database.

USE sql\_inventory;

SELECT \*

FROM sql\_store.order\_items oi

JOIN sql\_invetory.products p

ON oi.product\_id = p.product\_id

15.---- Self joins , you can also join a table within itself (let’s say you only want to find the manager in the employees list)

USE sql\_hr;

SELECT \*

FROM employees e

JOIN employees m

ON e.reports\_to = m.employee\_id

USE sql\_hr;

--- get each Column with a table name,

from the employees table you want to get the employee id and first name

from the manger table you want to get the manager

every column is prefixed with an alias (because all these columns exist in two tables)

* you should get the employee id, first name and the managers first name

SELECT

e.employee\_id,

e.first\_name,

m.first\_name

FROM employees e

JOIN employees m

ON e.reports\_to = m.employee\_id

16---- Joining multiple tables (use double join to join to tables)

USE sql\_store;

SELECT\*

FROM orders

JOIN customers

ON orders.cutomer\_id=cutomers.cutomer\_id

JOIN order\_stauses os

ON orders.status =os.order\_status\_id

--- example , you can also explicitly select few of the columns from the table

USE sql\_store;

SELECT

o.order\_id,

o.order\_date,

c.first\_name,

c.last\_name,

os.name AS status

FROM orders. o

JOIN customers c

ON orders.cutomer\_id=cutomers.cutomer\_id

JOIN order\_stauses os

ON orders.status =os.order\_status\_id

17.--- compound join conditions (

18.--- Implicit Join syntax ( you can write the same query)

this one is the same as

SELECT\*

FROM orders

JOIN customers

ON orders.cutomer\_id=cutomers.cutomer\_id

--- this one : ( this is called implicit join syntax)

SELECT\*

FROM orders O, customers c

WHERE orders.cutomer\_id=cutomers.cutomer\_id

OUTER JOIN

19.--- Outer join (to types of out joins left and right)

left join

--- when we use a left join all the records in customer c, are returned whether this condition is true or not.

SELECT\*

FROM customers c

LEFT JOIN orders o

ON c.customer\_id= o.customer\_id

ORDER BY c.customer\_id

right join

--- when we use a rightjoin all the records in customer c, are returned whether this condition is true or not.

(Remember to swap the customer with orders)

SELECT\*

FROM orders o

LEFT JOIN customer c

ON c.customer\_id= o.customer\_id

ORDER BY c.customer\_id

20. Outer join between multiple tables

21. Self outer join

--- what to get the employees and their manager .

USE sql\_hr ;

SELECT \*

e.employee\_id

e.first\_name

m.first\_name AS manager

FROM employee e

LEFT JOIN employee m

ON e.report\_to=m.emplyee\_id

22. The USING CLAUSE

SELECT \*

o.order\_id,

c.first\_name,

FROM orders. o

JOIN customers c

ON orders.cutomer\_id=cutomers.cutomer\_id

--- we can use USING (customer\_id) to short the ON condition

USING (customer\_id)

--- if the column name is exactly the same across these two tables , we can replace the on clause with the USING

23. Natural join (Joining to tables)

--- natural join are easier to code but they can be a little bit dangerous, because we are letting the database engine or guess the join, you do not have control it

SELECT

o.order\_id

c.first\_name

FROM orders o

NATURAL JOIN customers c

24.Cross joins

--- we use cross join to combine or join every record from the first table, with every record in a second table

SELECT \*

FROM customers c

CROSS JOIN products p

--- example

SELECT

sh. name AS shipper,

p.name AS product

FROM shippers sh

CROSS JOIN products p

ORDER BY sh.name

25. UNION

--- using the union operator, we can combine records for multiple queries

--- this combine

SELECT\*

order\_id,

order\_date,

“Active” AS status

FROM orders

WHERE order\_date>= “2019-01-01”

UNION

--- this returns the query with the label archived

SELECT\*

order\_id,

order\_date,

“Archive” AS status

FROM orders

WHERE order\_date <= “2019-01-01”

---- using the union operator, we can combine data from these two queries by placing in between

--- in this example both our queries are against the same table, but we can also have queries against different tables and then combine the result int one result set.

SELECT first\_name

FROM customers

UNION

SELECT name

FROM shippers

--- if you want to change the name of the column you can use AS

SELECT customer\_id,

first\_name,

points,

“Bronze” AS type

FROM customers

WHERE points < 2000

--- if you want to get more than one type , you place a UNION in between

--- if you want to order them by first name , you place ORDER BY first\_name

SELECT customer\_id,

first\_name,

points,

“Bronze” AS type

FROM customers

WHERE points < 2000

UNION

SELECT customer\_id,

first\_name,

points,

“Silver” AS type

FROM customers

WHERE points BETWEEN 2000 AND 3000

ORDER BY first\_name

26.Column Attributes (INSERT, UPDATE AND DELETE DATA)

click on the key , PK primary , NN not null , AI

27.Inserting a single row

--- Insert a row into a table

---DEFAULT generates a value

INSERT INTO customers

VALUES(DEFAULT, “Jhon” , “Smith”, “1990-01-01”, 11111111,)

--- after the table name we can optionally supply the stuff columns that you want

INSERT INTO customers (first\_name , last\_name, birth\_date, address, city, state)

VALUES(DEFAULT, “John” , “Smith”, “1990-01-01”, guruveien 10, Oslo, Oslo)

28.Inserting multiple rows in one go

---- shipper is the table and (name) is the name of the column we want to insert values into

--- values , here we insert multiple values in the row

INSERT INTO shipper (name)

VALUES (“shipper1”), (“shipper2”), (“shipper3”)

29.Inserting hierarchical rows

-how to insert data into multiple tables

30.Create a copy of the table

-- Vekselvirkende delspørringer

-- billigste vare i hver kategori, Alternativ 1:

-- resultatet blir 20 rader, men det finnes flere varer med samme pris

-- så det vi vet er at det finnes færre enn 20 billigste varer i hver kategori

SELECT Vare1.VNr, Vare1.Betegnelse, Vare1.KatNr, Vare1.Pris

FROM Vare AS Vare1

WHERE Vare1.Pris=

(SELECT MIN(Vare2.Pris)

FROM Vare AS Vare2

WHERE Vare1.KatNr=Vare2.KatNr)

;