



DROP DATABASE IF EXISTS `emp`;

CREATE DATABASE emp;

USE emp;

CREATE TABLE info

(

empid INT,

empfname VARCHAR(255),

age INT,

emailid VARCHAR(255)

);

INSERT INTO info(empid, empfname, age, emailid)

VALUES('1','ANKIT', '25', 'abc.com')

USE sql\_store; #first we need to select the database

SELECT \* FROM customers. # \* denotes every column

-- WHERE customer\_id = 1

ORDER BY first\_name

2.

SELECT first\_name, last\_name, points FROM customers.

#points is a new column which is given based on the customer data, we can change the value of this column using arithmetic operation

-- SELECT \* FROM customers

-- WHERE customer\_id = 1

ORDER BY first\_name

3.

SELECT first\_name, last\_name, points + 10 FROM customers

-- SELECT \* FROM customers

-- WHERE customer\_id = 1

ORDER BY first\_name

4.

USE sql\_store;

SELECT first\_name, last\_name, points+10, points \*10+100 AS discount\_factor FROM customers

#Using AS keyword, we are renaming the column points

-- SELECT \* FROM customers

-- WHERE customer\_id = 1

ORDER BY points DESC

#DESC denotes descending order

5.

UPDATE `sql\_store`.`customers` SET `state` = 'VA' WHERE (`customer\_id` = '1');

#UPDATES OUR DATA ON THE TABLE CUSTOMERS WHERE CUSTOMER ID = 1

6.

SELECT DISTINCT state FROM sql\_store.customers;

#this doesn’t return any duplicate values. Only returns unique values

7.

EXERCISE

There is a table products. You need to return the name, unit price and a new column NEW PRICE which will be 1.1 time the unit price

Unit\_price\*1.1 = new price

8.

Operators in sql

>

>=

<

<=

=

!=

SELECT \* FROM sql\_store.customers

WHERE points > 2500

SELECT \* FROM sql\_store.customers

-- WHERE points > 2500;

-- WHERE state != 'CO'

WHERE birth\_date > '1990-01-01'

-- WHERE birth\_date > '1990-01-01' AND points > 1000

WHERE birth\_date > '1990-01-01' OR points > 1000

WHERE birth\_date > '1990-01-01' OR points > 1000 AND state = 'VA'

8. exercise

From order\_items table, get the items for ORDER NO 6

Where the total price is greater than 30

SOL:

SELECT \* FROM order\_items

WHERE order\_id = 6 AND unit\_price \* quantity > 30

9. IN OPERATOR- when you want to use a list of values

-- WHERE state IN ('VA', 'FL', 'GA')

WHERE state NOT IN ('VA', 'FL', 'GA')

10.

Return products with quantity in stock equal to 49,38,72

11. BETWEEN OPERATOR

where points BETWEEN 1000 AND 3000

12. LIKE OPERATOR

WHERE last\_name LIKE 'b%'

WHERE last\_name LIKE '%b%'

WHERE last\_name LIKE '%y'

13. IS null operator – checks whether a place in table is empty/ null

WHERE phone IS NULL

14. ORDER BY first\_name

For ascending order

ORDER BY first\_name DESC

For descending order

15. sorting based on multiple values

ORDER BY state, first\_name

16. sorting states in descending order and name in ascending

ORDER BY state DESC, first\_name

17. exercise: on the table ORDER ITEMS where select items with order id = 2 and sort them based on their total price (no of items \* price per unit) in descending order

SELECT \*, quantity \* unit\_price AS total\_price FROM order\_items

WHERE order\_id = 2

ORDER BY quantity \* unit\_price DESC

18. OFFSET AND LIMIT

LIMIT 6, 3

#here offset = 6 and limit is 3

19. EXERCISE:

GET THE TOP # LOYAL CUSTOMERS (who have the maximum points)

ORDER BY points DESC

LIMIT 3

20. INNER JOIN

SELECT order\_id, first\_name, last\_name, orders.customer\_id FROM sql\_store.orders

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

#name of the column through which you want to join the tables

20. EXERCISE: display the order id, product id and product name by joining order items and products

SELECT \* FROM order\_items

JOIN products ON order\_items.product\_id = products.product\_id

21. joining if both the tables are in different databases(JOINING ACROSS DATABASES)

SELECT \* FROM sql\_store.order\_items o

JOIN sql\_inventory.products p ON o.product\_id = p.product\_id

22. In a database, we have multiple employees and there is given whom to report to. We are asked to find the name of the manager for each employee like who they have to report. Both the informations are given in **same table**.

USE sql\_hr;

SELECT e.employee\_id, e.first\_name, m.first\_name AS manager FROM sql\_hr.employees e

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

23. JOINING Multiple tables

SELECT o.order\_id, o.order\_date, c.first\_name, os.name AS status FROM sql\_store.orders o

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

JOIN order\_statuses os ON o.status = os.order\_status\_id

24. EXERCISE: join the tables CLIENTS, PAYMENT METHODS & PAYMENTS in SQL INVOICING database

USE sql\_invoicing;

SELECT \* FROM payments p

JOIN clients c ON p.client\_id = c.client\_id

JOIN payment\_methods pm ON p.payment\_method = pm.payment\_method\_id

25. JOINING MULTIPLE COLUMNS (compound join condition)

SELECT \* FROM order\_items oi

JOIN order\_items\_notes oin ON oi.order\_id = oin.order\_id AND oi.product\_id = oin.product\_id

26. IMPLICIT JOIN SYNTAX

SELECT \* FROM orders o , customers c

WHERE o.customer\_id = c.customer\_id

27. OUTER JOINS (customers who have an order placed are returned. Now if we want to view all the customers who don’t have an order, we use outer join)

Types of OUTER JOIN

1. Left join – we get all the customers from customer table whether the condition is true or not
2. Right join- we get all the customers from orders table whether the condition is true or not

SELECT c.customer\_id, o.order\_id, c.first\_name FROM customers c

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

***LEFT JOIN***

SELECT c.customer\_id, o.order\_id, c.first\_name FROM customers c

LEFT JOIN orders o ON c.customer\_id = o.customer\_id

#all the records on left tables are returned. Here left table is customers

***RIGHT JOIN***

SELECT c.customer\_id, o.order\_id, c.first\_name FROM customers c

RIGHT JOIN orders o ON c.customer\_id = o.customer\_id

#all the records on orders table are returned whether the condition is true or not

28. USING CLAUES(works if both the column names with which you are joining have the same name)

SELECT \* FROM orders o

JOIN customers c USING (customer\_id)

Or

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

29. UNIONS – combining rows from multiple tables

SELECT order\_id, order\_date, ‘ACTIVE’ AS status FROM orders

WHERE order\_date >= ‘2019-01-01’

UNION

SELECT order\_id, order\_date, ‘ARCHIEVE’ AS status FROM orders

WHERE order\_date < ‘2019-01-01’

30.different tables

SELECT first\_name from customers

UNION

SELECT name from shippers

31.this throws an error. The number of columns while working with unions must be same

SELECT first\_name, last\_name from customers

UNION

SELECT name from shippers

32. INSERTING. A DATA INTO TABLE

INSERT INTO customers

VALUES (DEFAULT, ‘ANKIT’, ‘P’, ‘1995-01-06’, NULL/DEFAULT, ‘address’, ‘city’, ‘state’,DEFAULT )

OR

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

VALUES ( ‘ANKIT’, ‘P’, ‘1995-01-06’, ‘address’, ‘city’, ‘state’)

33. INSERTING MULTIPLE ROWS

INSERT INTO shippers (name)

VALUES (‘shipper1’), (‘shipper2’), (‘shipper3’)

34. UPDATING A data

UPDATE invoices

SET payment\_total = 10, payment\_date = ‘2019-04-05’

WHERE invoice\_id = 1

35. SUBQUERY – a SELECT statement that is present within another select statement

UPDATE invoices

SET payment\_total = invoice\_total \* 5, payment\_date = due\_date

WHERE client\_id =

(SELECT client\_id FROM clients WHERE name = ‘Myworks’)

36. DELETING rows

DELETE FROM invoices WHERE invoice\_id = 1

36. using sub query for deleting rows

DELETE FROM invoices WHERE client\_id = (

SELECT \* FROM clients WHERE name = ‘Myworks’

)

37.



