1. SELECT * FROM Customers;

2. /* Some important commands:*/

   SELECT UPDATE DELETE INSERT INTO CREATE DATABASE ?   ALTER DATABASE CREATE TABLE ALTER TABLE DROP TABLE CREATE INDEX DROP INDEX

3. SELECT column1, column2, ... FROM table_name;

4. SELECT DISTINCT column1, column2,... FROM table_name;

5. SELECT COUNT(DISTINCT Country) FROM Customers;

6. SELECT Count(*) AS DistinctCountries FROM (SELECT DISTINCT Country FROM Customers);

7. SELECT column1, column2, ... FROM table_name WHERE condition;

   SELECT * FROM Customers WHERE Country='Mexico';

8. Operator Description = Equal <> Not equal. Note: In some versions of SQL this operator may be written as != > Greater than < Less than >= Greater than or equal <= Less than or equal BETWEEN Between a certain range LIKE Search for a pattern IN To specify multiple possible values for a column

9. SELECT column1, column2 FROM table_name WHERE condition1 AND condition2 AND condition3....;

   OR, WHERE NOT condition

   SELECT * FROM Customers WHERE Country='Germany' AND City='Berlin';

10. SELECT column1,column2, FROM table_name ORDER BY column1, column2, ... ASC|DESC;

    ## 默认ASC 如果不特别备注

11. ## insert values

    !!!!!!!!!!! INSERT INTO Customers (CustomerName, ContactName, Address, City, PostalCode, Country) VALUES ('Cardinal', 'Tom B. Erichsen', 'Skagen 21', 'Stavanger', '4006', 'Norway');

    ## which equals add one row at the botton of the

table with the above values

in each column.

12. ## The IS NOT NULL Operator

    SELECT CustomerName, ContactName, Address FROM Customers WHERE Address IS NOT NULL;

    SELECT CustomerName FROM Customers WHERE Address IS NULL;

13. UPDATE table_name SET column1 = value1, column2 = value2, ... WHERE condition;

    UPDATE Customers SET ContactName = 'Alfred Schmidt', City = 'Frankfurt' WHERE CustomerID = 1;

    ## UPDATE Customers

    ## SET ContactName='Juan'

    ## WHERE Country='Mexico'

14. DELETE FROM Customers WHERE CustomerName='Alfreds Futterkiste'; !!! FROM very important!!!!

15. SELECT column*name(s) FROM table*name WHERE condition LIMIT number;

    !!!!! !!! !! SELECT TOP 50 PERCENT * FROM CustomerName;

16. SELECT MIN(column*name) FROM table*name WHERE condition;

    MAX()

    SELECT MIN(Price) AS SmallestPrice FROM Products;

17. SELECT COUNT(column*name) FROM table*name WHERE condition;

    AVG() SUM() SELECT (*) FROM Products Price = 18;

18. LIKE Operator Description WHERE CustomerName LIKE 'a%' Finds any values that start with "a" WHERE CustomerName LIKE '%a' Finds any values that end with "a" WHERE CustomerName LIKE '%or%' Finds

any values that have "or" in any position WHERE CustomerName LIKE '*r%' Finds any values that have "r" in the second position WHERE CustomerName LIKE 'a*%_%' Finds any values that start with "a" and are at least 3 characters in length WHERE ContactName LIKE 'a%o' Finds any values that start with "a" and ends with "o"

！！！！！！！！ SELECT * FROM Customers WHERE City LIKE '[a-c]%';

SELECT * FROM Customers WHERE City LIKE '[!bsp]%';

19. SELECT column*name(s) FROM table*name WHERE column_name IN (SELECT STATEMENT);

Use the IN operator to select all the records where Country is either "Norway" or "France".

SELECT * FROM Customers 'France';

20. SELECT column*name(s) FROM table*name WHERE column_name BETWEEN value1 AND value2 ORDER BY ;

21. !!!!! SELECT column*name AS alias*name FROM table_name;

    SELECT CustomerID AS ID, CustomerName AS Customer FROM Customers;

    SELECT CustomerName, Address + ', ' + PostalCode + ' ' + City + ', ' + Country AS Address

22. SQL JOINS !!!!!

    Here are the different types of the JOINs in SQL:

    (INNER) JOIN: Returns records that have matching values in both tables LEFT (OUTER) JOIN: Return all records from the left table, and the matched records from the right table RIGHT (OUTER) JOIN: Return all records from the right table, and the matched records from the left table FULL (OUTER) JOIN: Return all records when there is a match in either left or right table

    LEFT JOIN 就是左边的都在的 right join 就是右边的都在的 OUTER

    full outer join 就是都存在所有的并集交际都放在一起

    SELECT * FROM Orders LEFT JOIN Customers ON orders.CustomerID = Customers.CustomerID;

!!!!!之后继续复习join

1. SELECT column*name(s) FROM table1 INNER JOIN table2 ON table1.column*name = table2.column_name;

2. SELECT A.CustomerName AS CustomerName1, B.CustomerName AS CustomerName2, A.City FROM Customers A, Customers B WHERE A.CustomerID <> B.CustomerID AND A.City = B.City ORDER BY A.City;

   ！！！！ . 很重要 A. SS

3. SELECT column*name(s) FROM table1 UNION SELECT column*name(s) FROM table2;

   UNION ALL # all duplicates values.

   SELECT City, Country FROM Customers WHERE Country='Germany' UNION ALL SELECT City, Country FROM Suppliers WHERE Country='Germany' ORDER BY City;

   ```
   City    Country
   Aachen  Germany
   Berlin  Germany
   Berlin  Germany
   Brandenburg Germany
   Cunewalde   Germany
   Cuxhaven    Germany
   Frankfurt   Germany
   Frankfurt a.M.  Germany
   ```

4. ## group by

   SELECT column*name(s) FROM table*name WHERE condition GROUP BY column*name(s) ORDER BY column*name(s);

   SELECT Shippers.ShipperName, COUNT(Orders.OrderID) AS NumberOfOrders FROM Orders LEFT JOIN Shippers
   ON Orders.ShipperID = Shippers.ShipperID GROUP BY ShipperName;

   SELECT COUNT(CustomerID), Country FROM Customers GROUP BY Country HAVING COUNT(CustomerID) > 5 ORDER BY COUNT(CustomerID) DESC;

5. SELECT column*name(s) FROM table*name WHERE condition GROUP BY column*name(s) HAVING condition ORDER BY column*name(s);

6. **count function 一定要用括号里给 范围**

7. !!!!! SELECT column*name(s) FROM table*name WHERE EXISTS (SELECT ProductName FROM ProductName FROM Products WHERE SuppliersID = Suppliers.SuppliersID AND Price < 20)

8. !! ```The ANY and ALL operators are used with a WHERE or HAVING clause.

   The ANY operator returns true if any of the subquery values meet the condition.

   The ALL operator returns true if all of the subquery values meet the condition.

   ```
   SELECT column_name(s)
   FROM table_name
   WHERE column_name operator ANY/ALL
   (SELECT column_name FROM table_name WHERE condition);
   # return 还是table 只是 会根据这个条件return table
   ```

1. The following SQL statement creates a backup copy of Customers:

   SELECT * INTO CustomersBackup2017 FROM Customers;

2. !!!! INSERT INTO table2 SELECT * FROM table1 WHERE condition;

3. !!! # case

   CASE WHEN condition1 THEN result1 WHEN condition2 THEN result2 WHEN conditionN THEN resultN ELSE result END;

   SELECT OrderID, Quantity, CASE WHEN Quantity > 30 THEN "The quantity is greater than 30" WHEN Quantity = 30 THEN "The quantity is 30" ELSE "The quantity is under 30" END AS QuantityText FROM OrderDetails;

   !!!!!!! SELECT CustomerName, City, Country FROM Customers ORDER BY (CASE WHEN City IS NULL THEN Country ELSE City END);

   SELECT CustomerName, City, Country FROM Customers ORDER BY (CASE WHEN CITY IS NULL THEN Country ELSE CustomerName END);

4. SELECT ProductName, UnitPrice * (UnitsInStock + IFNULL(UnitsOnOrder, 0)) FROM Products

   SELECT ProductName, UnitPrice * (UnitsInStock + COALESCE(UnitsOnOrder, 0)) FROM Products

SELECT ProductName, UnitPrice * (UnitsInStock + ISNULL(UnitsOnOrder, 0)) FROM Products

```
CREATE PROCEDURE SelectAllCustomers @City nvarchar(30)          ```
AS
SELECT * FROM Customers WHERE City = @City
GO;
```

1. DATABASE

CREATE DATABASE DROP DATABASE

1. SQL is Structured Query Language, which is a computer language for storing, manipulating and retrieving data stored in relational database.

2.
3. There are various components included in the process.

4. These components are Query Dispatcher, Optimization Engines, Classic Query Engine and SQL Query Engine, etc.

5. query engine handles all non-SQL queries, but SQL query engine wont handle logical files.

6. CREATE TABLE CUSTOMERS(ID INT NOT NULL, NOT NULL, NOT NULL, NAME VARCHAR (20) AGE INT ADDRESS CHAR (25) , SALARY DECIMAL (18, 2), PRIMARY KEY (ID));

7. ALTER TABLE CustomerName;

8. CREATE TABLE Customers();

9. SQL IN Clause: SQL BETWEEN Clause: SQL ORDER BY Clause: SELECT column1, column2....columnN FROM table*name WHERE column*name IN (val-1, val-2,...val-N);

10. SQL LIKE Clause: SELECT column1, column2....columnN FROM table*name WHERE column*name LIKE { PATTERN }; SELECT column1, column2....columnN FROM table*name WHERE CONDITION ORDER BY column*name {ASC|DESC}; SQL GROUP BY Clause: SELECT SUM(column*name) FROM table*name WHERE CONDITION GROUP BY column*name; SQL COUNT Clause: SQL HAVING Clause: SQL CREATE TABLE Statement: CREATE TABLE table*name(TUTORIALS POINT Simply Easy Learning

SELECT COUNT(column*name) FROM table*name WHERE CONDITION;

SELECT SUM(column*name) FROM table*name WHERE CONDITION GROUP BY column_name HAVING (arithematic function condition)

1. SQL CREATE TABLE Statement: CREATE TABLE table_name(column1 datatype, column2 datatype,

column3 datatype, ... columnN datatype, PRIMARY KEY(one or more columns));

SQL CREATE INDEX Statement: SQL DROP INDEX Statement: SQL DESC Statement: SQL ALTER TABLE Statement: ALTER TABLE table*name {ADD|DROP|MODIFY} column*name {data*ype}; SQL ALTER TABLE Statement (Rename): ALTER TABLE table*name RENAME TO new*table*name; SQL INSERT INTO Statement: SQL UPDATE Statement:

2. DELETE SS FROM TABLE_NAME WHERE

3.

SQL Logical Operators: Here is a list of all the logical operators available in SQL.

Operator

Description

ALL

The ALL operator is used to compare a value to all values in another value set.

AND

The AND operator allows the existence of multiple conditions in an SQL statements WHERE clause.

The ANY operator is used to compare a value to any applicable value in the list according to the condition.

ANY

The BETWEEN operator is used to search for values that are within a set of values, given the minimum value and the maximum value.

BETWEEN

The EXISTS operator is used to search for the presence of a row in a specified table that meets certain criteria.

EXISTS

IN

The IN operator is used to compare a value to a list of literal values that have been specified.

LIKE

The LIKE operator is used to compare a value to similar values using wildcard operators.

The NOT operator reverses the meaning of the logical operator with which it is used. Eg: NOT EXISTS, NOT BETWEEN, NOT IN, etc. This is a negate operator.

NOT

OR

The OR operator is used to combine multiple conditions in an SQL statements WHERE clause.

IS NULL

The NULL operator is used to compare a value with a NULL value.

UNIQUE

The UNIQUE operator searches every row of a specified table for uniqueness (no duplicates).

1. There are several built-in functions like avg(), sum(), count(), etc., to perform what is known as aggregate data calculations against a table or a specific table column.

SQL> SELECT COUNT(*) AS "RECORDS"

FROM CUSTOMERS; +---------+ | RECORDS | +---------+ |7| +---------+ 1 row in set (0.00 sec)

1. SQL> SELECT GETDATE();;

46. SELECT CURRENT_TIMESTAMP;

1. SQL> CREATE DATABASE testDB; CHAPTER The SQL CREATE DATABASE statement is used to create new SQL database. Syntax:

   Make sure you have admin privilege before creating any database. Once a database is created, you can check it in the list of databases as follows: SQL> SHOW DATABASES; +--------------------+ | Database | +--------------------+ | information_schema | | AMROOD | | TUTORIALSPOINT | | mysql | | orig | | test | | testDB | +--------------------+ 7 rows in set (0.00 sec)

2. SELECT film_id, title FROM film WHERE title LIKE 'Ch%';

49.

SELECT column1, column2, columnN FROM table_name WHERE [condition1] OR [condition2]...OR [conditionN]

1.

SQL> UPDATE CUSTOMERS SET ADDRESS = 'Pune', SALARY = 1000.00;

1. Following is an example, which would DELETE a customer, whose ID is 6: Now, CUSTOMERS table would have the following records: TUTORIALS POINT Simply Easy Learning SQL> DELETE FROM CUSTOMERS

2.

SELECT current_timestamp;

1. SQL LIKE clause is used to compare a value to similar values using wildcard operators. There are two wildcards used in conjunction with the LIKE operator: The percent sign (%) The underscore (_) The percent sign represents zero, one, r multiple characters. The underscor e represents a single number or c haracter. The symbols can be used in combinations.

case sensitive

1. /* */

2.

SELECT * FROM film WHERE title LIKE 'Ac_ ____fing%'

1.

SELECT * FROM film LIMIT 3 OFFSET 2;

SELECT * FROM film ORDER BY film_id DESC LIMIT 3 OFFSET 2

1.

SELECT * FROM film;

SELECT film*id, title, rental*rate FROM film WHERE rental_rate< 9

ORDER BY rental_rate DESC

1. SELECT SUM(rental*rate) AS rental*rate*sum FROM film GROUP BY rental*rate ;

SELECT * FROM CUSTOMERS ORDER BY (CASE ADDRESS WHEN 'DELHI' WHEN 'BHOPAL' WHEN 'KOTA' WHEN 'AHMADABAD' THEN 4 WHEN 'MP' THEN 5 ELSE 100 END) ASC, ADDRESS DESC;

1.

SELECT rental*rate FROM film ORDER BY (CASE rental*rate WHEN 2.99 THEN 1 WHEN 4.99 THEN 2 ELSE 100 END) DESC, rental_rate ASC;

CREATE TABLE CUSTOMERS(ID INT NOT NULL, NOT NULL, NOT NULL, NAME VARCHAR (20) AGE INT ADDRESS CHAR (25) , SALARY DECIMAL (18, 2) DEFAULT 5000.00, PRIMARY KEY (ID));

CREATE TABLE CUSTOMERS(ID INT NOT NULL, NOT NULL, NOT NULL, NAME VARCHAR (20) AGE INT ADDRESS CHAR (25) , SALARY DECIMAL (18, 2) DEFAULT 5000.00, PRIMARY KEY (ID));

1. CARTESIAN JOIN: returns the Cartesian product of the sets of

2. SELECT customer.customer*id AS CAO, customer.store*id AS NI , film.film*id AS MA FROM customer INNER JOIN film ON customer.customer*id = film.film_id;

3.

SELECT table1.column1, table2.column2... FROM table1 LEFT JOIN table2 ON table1.common*filed = table2.common*field;

1. The basic syntax of UNION is as follows: Here given condition could be any given expression based on your requirement. Example: Consider the following two tables, (a) CUSTOMERS table is as follows: CHAPTER 27 The SQL UNION clause/operator is used to combine the results of two or more SELECT statements without returning any duplicate rows. To use UNION, each SELECT must have the same number of columns selected, the same number of column expressions, the same data type, and have them in the same order, but they do not have to be the same length. SELECT column1 [, column2] FROM table1 [, table2] [WHERE condition] UNION SELECT column1 [, column2] FROM table1 [, table2] [WHERE condition]

2. SQL> SELECT ID, NAME, AMOUNT, DATE FROM CUSTOMERS LEFT JOIN ORDERS ON CUSTOMERS.ID = ORDERS.CUSTOMER*ID UNION SELECT ID, NAME, AMOUNT, DATE FROM CUSTOMERS RIGHT JOIN ORDERS ON CUSTOMERS.ID = ORDERS.CUSTOMER*ID;

UNION ALL

SQL INTERSECT Clause: is used to combine two SELECT statements, but returns rows only from the first SELECT statement that are identical to a row in the second SELECT statement. SQL EXCEPT Clause : combines two SELECT statements and returns rows from the first SELECT statement that are not returned by the second SELECT statemen

1. SELECT column1 [, column2] FROM table1 [, table2] [WHERE condition]

EXCEPT SELECT column1 [, column2] FROM table1 [, table2] [WHERE condition]

1.

SELECT * FROM customer;

SELECT customer.customer*id AS CAO, customer.store*id AS NI , film.film*id AS MA FROM customer LEFT JOIN film ON customer.customer*id = film.film_id;

EXCEPT

SELECT customer.customer*id AS CAO, customer.store*id AS NI , film.film*id AS MA FROM customer RIGHT JOIN film ON customer.customer*id = film.film_id;

Indexes should not be used on small tables. □ Tables that have frequent, large batch update or insert operations. □ Indexes should not be used on columns that contain a high number of NULL values. □ Columns that are frequently manipulated should not be indexed.

1.

DROP INDEX pig;

68.

SELECT column1, column2 FROM table1, table2 WHERE [conditions] GROUP BY column1, column2 HAVING [conditions] ORDER BY column1, column2

69.

ADDDATE()

Adds dates

ADDTIME()

Adds time

CONVERT_TZ()

Converts from one timezone to another

CURDA TE()

Returns the current date

CURRENT*DATE(), CURRENT*DATE

Synonyms for CURDATE()

CURRENT*TIME(), CURRENT*TIME

Synonyms for CURTIME()

CURRENT*TIMESTAMP(), CURRENT*TIMESTAMP

Synonyms for NOW()

CURTIME()

Returns the current time

DATE_ADD()

Adds two dates

DATE_FORMAT()

Formats date as specified

DATE_SUB()

Subtracts two dates

DATE()

Extracts the date part of a date or datetime expression

DATEDIFF()

Subtracts two dates

DAY()

Synonym for DAYOFMONTH()

DAYNAME()

Returns the name of the weekday

DAYOFMONTH()

Returns the day of the month (1-31)

DAYOFWEEK()

Returns the weekday index of the argument

TUTORIALS POINT Simply Easy Learning CHAPTER 37

70.

mysql> SELECT MAKEDATE(2001,31), MAKEDATE(2001,32); +---+ | MAKEDATE(2001,31), MAKEDATE(2001,32) | +---+ | '2001-01-31', '2001-02-01' | +---+ 1 row in set (0.00 sec)