|  |  |  |  |
| --- | --- | --- | --- |
| NO | MACAM-MACAM SQL | PENGERTIIAN/FUNGSI | CONTOH |
| 1 | SQL HOME | Bahasa standar untuk mengakses data | SELECT \* FROM Customers; |
| 2 | SQL INTRO | Bahasa standar untuk mengakses dan memanipulasi data base | SELECT \* FROM Customers; |
| 3 | SQL SYNTAX | Sebuah data base yang paling sering mengandung satu atau lebih table | SELECT \* FROM Customers; |
| 4 | SQL SELEC | Digunakan untuk memilih data dari database | SELECT CustomerName,City FROM Customers; |
| 5 | SQL SELEC DISTINCT | Digunakan untuk kembali hanya berada nilai (berbeda) | SELECT DISTINCT City FROM Customers; |
| 6 | SQL WHERE,AND,OR,NOT | Where digunakan untuk menyaring data, and operator yang digunakan untuk menyaring catatan berdasarkan lebih dari satu kondisi | (Where) SELECT \* FROM Customers WHERE Country='Mexico';  (AND) SELECT \* FROM Customers WHERE Country='Germany' AND City='Berlin';  (OR) SELECT \* FROM Customers WHERE City='Berlin' OR City='Munchen';  NOT, SELECT nis, nama FROM siswa WHERE NOT alamat='Jakarta' |
| 7 | QSL ORDE BY | Digunakan untuk mengurut hasil-set | SELECT \* FROM Customers ORDER BY Country; |
| 8 | INSERT INTO | Digunakan untuk menyisipkan catatan baru dalam sebuah table | INSERT INTO Customers (CustomerName, ContactName, Address, City, PostalCode, Country) VALUES ('Cardinal','Tom B. Erichsen','Skagen 21','Stavanger','4006','Norway'); |
| 9 | SQL NULL VALUES | Mewakili data yang hilang tidak diketahui | SELECT LastName,FirstName,Address FROM Persons WHERE Address IS NULL |

|  |  |  |  |
| --- | --- | --- | --- |
| 10 | SQL DELETE | Digunakan untuk menghapus baris dalam sebuah table | DELETE FROM Customers WHERE CustomerName='Alfreds Futterkiste' AND ContactName='Maria Anders'; |
| 11 | SQL SELECT TOP | Digunakan untuk menentukan jumlah record untuk kembali | SELECT \* FROM Persons LIMIT 5; |
| 12 | SQL MIN, AND MAX | (MIN) mengembalikan nilai terkecil dari kolom yang dipilih  (MAX) mengembalikan nilai terbesar dari kolom yang terpilih | (MIN) SELECT MIN(Price) AS SmallestOrderPrice FROM Products;  (MAX) SELECT MAX(Price) AS HighestPrice FROM Products; |
| 13 | SQL COUNT,AVG,SUM | (COUNT) mengembalikan fungsi jumlahbaris yang cocok dengan kiteria tertentu  (AVG) mengembalikan rata-rata dari kolom numeric  (SUM) mengembalikan fungsi total dari kolom numeric | (COUNT)SELECT COUNT(CustomerID) AS OrdersFromCustomerID7 FROM Orders WHERE CustomerID=7;  (AVG) SELECT AVG(Price) AS PriceAverage FROM Products;  (SUM) SELECT SUM(Quantity) AS TotalItemsOrdered FROM OrderDetails; |
| 14 | SQL LIKE | Digunakan untuk mencari polayang ditentukan dalam kolom | SELECT \* FROM Customers WHERE City LIKE 's%'; |
| 15 | SQL WILDCARDS | Digunakan untuk menggantikan krakter lain (s) dalam sebuah string | SELECT \* FROM Customers WHERE City LIKE '\_erlin'; |
| 16 | SQL IN | Memungkinkan anda untuk menentukan beberapa nilai dalam klasual MANA | SELECT \* FROM Customers WHERE City IN ('Paris','London'); |
| 17 | SQL BETWEEN | Digunakan untuk memilih nilai-nilai dalam kisaran, nilai-nilai dapat angka, teks, atau tunggal | SELECT \* FROM Products WHERE Price NOT BETWEEN 10 AND 20; |
| 18 | SQL ALIASES | Digunakan untuk sementara mengubah nama table atau kolom pos | SELECT CustomerName AS Customer, ContactName AS [Contact Person] FROM Customers; |

|  |  |  |  |
| --- | --- | --- | --- |
| 19 | SQL JOINS | Digunakan untuk menggabungkan bari dari dua atau lebih | SELECT Orders.OrderID, Customers.CustomerName, Orders.OrderDate FROM Orders INNER JOIN Customers ON Orders.CustomerID=Customers.CustomerID; |
| 20 | SQL INNER JOIN | Untuk memilih semua baris dari kedua table selama ada kecocokan Antara kolom dikedua table | SELECT Customers.CustomerName, Orders.OrderID FROM Customers INNER JOIN Orders ON Customers.CustomerID=Orders.CustomerID ORDER BY Customers.CustomerName; |
| 21 | SQL LEFT JOIN | Kembali kunci semua baris dari table kiri dengan baris yang cocok ditable kanan hasilnya adalah null disisi kanan ketika tidak ada pertandingan | SELECT Customers.CustomerName, Orders.OrderID FROM Customers LEFT JOIN Orders ON Customers.CustomerID=Orders.CustomerID ORDER BY Customers.CustomerName; |
| 22 | SQL RIGHT JOIN | kembali kunci semua baris dari tabel kanan (table2), dengan baris yang cocok di tabel kiri (table1). Hasilnya adalah NULL di sisi kiri ketika tidak ada pertandingan. | SELECT Orders.OrderID, Employees.FirstName FROM Orders RIGHT JOIN Employees ON Orders.EmployeeID=Employees.EmployeeID ORDER BY Orders.OrderID; |
| 23 | SQL FULL JOIN | pengembalian kunci semua baris dari tabel kiri (Table1) dan dari tabel kanan (table2).  The FULL Outer BERGABUNG kata kunci menggabungkan hasil dari kedua KIRI dan KANAN bergabung. | SELECT Customers.CustomerName, Orders.OrderID FROM Customers FULL OUTER JOIN Orders ON Customers.CustomerID=Orders.CustomerID ORDER BY Customers.CustomerName; |

|  |  |  |  |
| --- | --- | --- | --- |
| 24 | SQL SELF JOIN | Gabungan regular tetapi table bergabung dengan dirinya sendiri | SELECT A.NamaCustomer AS NamaCustomer1, B.NamaCustomer AS NamaCustomer2, A.Kota  FROM Customers A, Customers WHERE A.IdCustomer <> B.IdCustomer AND A.Kota = B.Kota ORDER BY A.Kota; |
| 25 | SQL UNION | Menggabungkan hasil dari dua atau lebih select pernyataan | SELECT City FROM Customers UNION SELECT City FROM Suppliers ORDER BY City; |
| 26 | SQL GROUP BY | Digunakan untuk mengelompokkan data dengan kiteria tertentu | SELECT column\_name,aggregate\_function(column\_name) FROM table\_name WHERE column\_name operator value GROUP BY column\_name |
| 27 | SQL HAVING | ditambahkan ke dalam SQL karena keyword MANA tidak dapat digunakan dengan fungsi agregat. | SELECT Employees.LastName, COUNT(Orders.OrderID) AS NumberOfOrders FROM (Orders INNER JOIN Employees ON Orders.EmployeeID=Employees.EmployeeID) GROUP BY LastName HAVING COUNT(Orders.OrderID) > 10; |
| 28 | SQL EXISTS | digunakan dengan cara itu, tetapi sebenarnya itu dimaksudkan sebagai cara untuk menghindari penghitungan | SELECT \*  FROM Orders o  WHERE EXISTS (  SELECT \*  FROM Products p  WHERE p.ProductNumber = o.ProductNumber) |
| 29 | SQL ANY, ALL | ANY dan ALL dapat digunakan dengan klausa WHERE atau HAVING. | SELECT NamaProduk  FROM Products  WHERE IdProduct = ANY (SELECT IdProduct FROM OrderDetails WHERE Quantity = 12); |
| 30 | SQL SELECT INTO | SELECT INTO memilih data dari satu meja dan memasukkan ke dalam tabel baru. | SELECT column\_name(s)  INTO newtable [IN externaldb ]  FROM table1; |

|  |  |  |  |
| --- | --- | --- | --- |
| 31 | SQL INSERT INTO SELECT | INSERT INTO SELECT salinan pernyataan data dari satu meja dan memasukkan ke dalam tabel yang ada. | INSERT INTO Customers (CustomerName, Country)  SELECT SupplierName, Country FROM Suppliers; |
| 32 | SQL CASE | Case statement dalam SQL mengembalikan nilai pada kondisi tertentu. | CASE <Case\_Expression>  WHEN Value\_1 THEN Statement\_1  WHEN Value\_2 THEN Statement\_2  .  .  WHEN Value\_N THEN Statement\_N  [ELSE Statement\_Else]  END AS [ALIAS\_NAME] |
| 33 | SQL NULL FUNCATIONS | digunakan untuk menentukan bagaimana kita ingin memperlakukan nilai NULL. | SELECT ProductName,UnitPrice\*(UnitsInStock+ISNULL(UnitsOnOrder,0)) FROM Products |
| 34 | SQL STORED PROCEDURES | SQL yang disiapkan untuk dapat di simpan, sehingga kode tersebut dapat digunakan kembali berulang kali. | CREATE PROCEDURE SelectAllCustomers  AS  SELECT \* FROM Customers  GO; |
| 35 | SQL COMMENT | digunakan untuk menjelaskan bagian dari pernyataan SQL atau untuk mencegah eksekusi pernyataan SQL. | --Select all:  SELECT \* FROM Customers; |
| 36 | SQL OPERATORS | Operator ini digunakan untuk menentukan kondisi dalam pernyataan SQL dan berfungsi sebagai konjungsi untuk beberapa kondisi dalam sebuah pernyataan. | SELECT \* FROM Produk WHERE Harga= 18; |
| 37 | SQL CREATE DB | CREATE DATABASE pernyataan digunakan untuk membuat database. | CREATE DATABASE my\_db; |

|  |  |  |  |
| --- | --- | --- | --- |
| 38 | SQL DROP DB | DROP DATABASE dapat digunakan untuk menjatuhkan database SQL yang ada. | DROP DATABASE databasename; |
| 39 | SQL BACKUP DB | Backup dan restore database merupakan hal mendasar yang sangat penting untuk di pahami. | SE coba\_backup\_restore;  GO  BACKUP DATABASE coba\_backup\_restore  TO DISK = 'D:\BACKUP\coba\_backup\_restore.bak'  WITH NOFORMAT, NOINIT, NAME = N'Full Backup of coba\_backup\_restore DB', SKIP, NOREWIND, NOUNLOAD, STATS = 10  GO |
| 40 | SQL CREATE TABLE | CREATE TABLE pernyataan digunakan untuk membuat tabel dalam database.  Tabel tersebut akan disusun dalam baris dan kolom; dan setiap tabel harus memiliki nama. | CREATE TABLE Persons  (  PersonID int,  LastName varchar(255),  FirstName varchar(255),  Address varchar(255),  City varchar(255)  ); |
| 41 | SQL DROP TABLE | DROP TABLE dapat digunakan untuk menghapus seluruh data tabel yang ada dalam database. | TRUNCATE TABLE table\_name; |
| 42 | SQL ALTER TABLE | digunakan sebagai fungsi menambahkan atau menyisipkan kolom pada sebuah table di database. | ALTER TABLE <nama\_table>  ADD <nama\_kolom> <tipe\_data>; |
| 43 | SQL CONSTRAINTS | digunakan untuk membatasi jenis data yang dapat masuk ke tabel. Constraint dapat ditentukan ketika tabel dibuat (dengan pernyataan CREATE TABLE) atau setelah tabel dibuat (dengan pernyataan ALTER TABLE). | CREATE TABLE Persons(  P\_Id int NOT NULL,  LastName varchar(255) NOT NULL,  FirstName varchar(255),  Address varchar(255),  City varchar(255)  ) |

|  |  |  |  |
| --- | --- | --- | --- |
| 44 | SQL NOT NULL | Kendala NOT NULL memaksa kolom untuk tidak menerima nilai NULL.  Kendala NOT NULL memaksa field untuk selalu mengandung nilai. Ini berarti bahwa Anda tidak dapat menyisipkan catatan baru, atau memperbarui rekor tanpa menambahkan nilai untuk bidang ini. | CREATE TABLE PersonsNotNull  (  P\_Id int NOT NULL,  LastName varchar(255) NOT NULL,  FirstName varchar(255),  Address varchar(255),  City varchar(255)  ) |
| 45 | SQL UNIQUE | Kendala UNIK unik mengidentifikasi setiap record dalam tabel database.  The UNIK dan PRIMARY KEY kendala baik memberikan jaminan untuk keunikan untuk kolom atau set kolom.  Sebuah KEY constraint PRIMARY otomatis memiliki kendala UNIK didefinisikan di atasnya. | CREATE TABLE Persons  (  P\_Id int NOT NULL,  LastName varchar(255) NOT NULL,  FirstName varchar(255),  Address varchar(255),  City varchar(255),  UNIQUE (P\_Id)  ) |
| 46 | SQL PRIMARY KEY | RIMARY KEY kendala unik mengidentifikasi setiap record dalam tabel database.  kunci primer harus mengandung nilai-nilai UNIK. | CREATE TABLE Persons  (  P\_Id int NOT NULL PRIMARY KEY,  LastName varchar(255) NOT NULL,  FirstName varchar(255),  Address varchar(255),  City varchar(255)  ) |
| 47 | SQL FOREIGEN KEY | KUNCI ASING di satu meja poin untuk PRIMARY KEY di meja lain. | CREATE TABLE Orders  (  O\_Id int NOT NULL,  OrderNo int NOT NULL,  P\_Id int,  PRIMARY KEY (O\_Id),  FOREIGN KEY (P\_Id) REFERENCES Persons(P\_Id)  ) |
| 48 | SQL CHECK | digunakan untuk membatasi rentang nilai yang dapat ditempatkan dalam kolom. | CREATE TABLE Persons  (  P\_Id int NOT NULL,  LastName varchar(255) NOT NULL,  FirstName varchar(255),  Address varchar(255),  City varchar(255),  CHECK (P\_Id>0)  ) |
| 49 | SQL DEFAUL | DEFAULT kendala digunakan untuk memasukkan nilai default ke kolom.  Nilai default akan ditambahkan ke semua catatan baru, jika tidak ada nilai lain yang ditentukan. | CREATE TABLE Persons  (  P\_Id int NOT NULL,  LastName varchar(255) NOT NULL,  FirstName varchar(255),  Address varchar(255),  City varchar(255) DEFAULT 'Sandnes'  ) |
| 50 | SQL INDEX | Index adalah objek pada MySQL yang berisi data yang terurut - dari nilai-nilai pada satu atau lebih field dalam suatu table. | SELECT t1.\*, m1.nama\_produk, m2.nama\_cabang FROM  tr\_penjualan t1  JOIN ms\_produk m1 ON t1.kode\_produk = m1.kode\_produk  JOIN ms\_cabang m2 ON t1.kode\_cabang = m2.kode\_cabang  LIMIT 0, 1000000; |
| 51 | SQL AUTO INCREMENT | Sangat sering kita ingin nilai field kunci utama yang akan dibuat secara otomatis setiap kali rekor baru dimasukkan.  Kami ingin menciptakan medan auto-increment dalam sebuah tabel. | CREATE TABLE Persons  (  ID int NOT NULL AUTO\_INCREMENT,  LastName varchar(255) NOT NULL,  FirstName varchar(255),  Address varchar(255),  City varchar(255),  PRIMARY KEY (ID)  ) |

|  |  |  |  |
| --- | --- | --- | --- |
| 52 | SQL DATES | Bagian yang paling sulit ketika bekerja dengan tanggal adalah untuk memastikan bahwa format tanggal Anda mencoba | SELECT \* FROM Orders WHERE TanggalPesanan='2012-12-12' |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| 52 | SQL DATES | Bagian yang paling sulit ketika bekerja dengan tanggal adalah untuk memastikan bahwa format tanggal Anda mencoba untuk memasukkan, sesuai dengan format kolom tanggal dalam database. | SELECT \* FROM Orders WHERE OrderDate='2008-11-11’ |
| 53 | SQL VIEWS | Pandangan adalah tabel virtual.  Bab ini menunjukkan cara membuat, memperbarui, dan menghapus tampilan. | CREATE VIEW [Current Product List] AS SELECT ProductID,ProductName FROM Products WHERE Discontinued=No |
| 54 | SQL INJECTION | injection atau biasa yang dikenal dengan sebutan SQLi adalah suatu teknik penyerangan web dengan menggunakan kode SQL (Structured Query Language) yang berbahaya untuk memanipulasi database. Tindakan ini termasuk kedalam kategori hacking, dimana penyerang akan mencari celah keamanan dari suatu website | elect \* from User where userName = ‘joko’ AND password = ’adaaja' |
| 55 | SQL HOSTING | Jika Anda ingin situs web Anda untuk dapat menyimpan dan mengambil data dari database, web server Anda harus memiliki akses ke database-sistem yang menggunakan bahasa SQL.  Jika server web Anda di-host oleh Internet Service Provider (ISP), Anda akan harus mencari SQL hosting. | - |

|  |  |  |  |
| --- | --- | --- | --- |
| 56 | SQL DATE TYPE | Setiap kolom dalam tabel database diperlukan untuk memiliki nama dan tipe data.  pengembang SQL harus memutuskan apa jenis data akan disimpan di dalam setiap kolom tabel saat membuat tabel SQL. Tipe data adalah label dan pedoman bagi SQL untuk memahami apa jenis data diharapkan dalam setiap kolom, dan juga mengidentifikasi bagaimana SQL akan berinteraksi dengan data yang tersimpan. | CREATE TABLE `user`(  `id\_user` CHAR(5) PRIMARY KEY  `nama\_user` VARCHAR(25  `address\_user` TEXT,  `birthday\_user` DATE,  `zipcode\_user` INT(5),  gender\_user` ENUM('male','female'),  `country\_user` INT(3),  `photo\_user` TEXT,  `email\_user` VARCHAR(255)  ); |
| 57 | SQL KEY WORDS | dapat digunakan untuk menentukan beberapa nilai dalam klausa WHERE. | SELECT \* FROM Customers  WHERE Negara IN ('Jerman', 'Prancis', 'Inggris Raya'); |
| 58 | MYSQL FUNCATION | ungsi mengembalikan nilai pertama dari kolom yang dipilih. | SELECT TOP 1 CustomerName FROM Customers  ORDER BY CustomerID ASC; |
| 59 | SQL SERVER FUNCATION | Kelompok fungsi sistem adalah kelompok fungsi yang memberikan informasi tentang pemakaian server database oleh pemakai. | CREATE FUNCTION [dbo].[test] ( @X INT, @Y INT )  RETURNS INT  AS  BEGIN  RETURN @X + @Y  END  GO |

|  |  |  |  |
| --- | --- | --- | --- |
| 60 | SQL ACCESS FUNCATION | Access fuction mengabaikan Hentian baris dalam pernyataan SQL. Namun, pertimbangkan menggunakan garis untuk setiap klausa untuk membantu menyempurnakan keterbacaan pernyataan SQL Anda untuk diri Anda sendiri dan orang lain. | SELECT [E-mail Address], Company  FROM Contacts  WHERE City="Seattle" |
| 61 | SQL QUICK REF | Isi dari semua SQL yang ada diatas. | - |