|  |  |
| --- | --- |
| **General Info for SQL Programming**   * String datatypes are enclosed in single quotes (E.G. where fname = ‘Smith’) * Integer datatypes are not enclosed in quotes (E.G. where age = 21) * Declaring variables – must have ‘@’ in front of name followed by datatype (E.G. Declare @strname varchar(75)) | **Tips**   * Helpful site for connection strings: [www.connectionstrings.com](http://www.connectionstrings.com) * Write complex queries one step at a time, iteratively. * Always test your code before handing off your scripts * Save your scripts for reuse * Always backup your database or save a copy of your table before modifying the table or data within it |
| Entities - person, place, object, event, concept corresponds to a row in a table  Associate Entities- It serves to link other entities together in a many-to-many relationship  Attributes- Properties or characteristics of an entity or relationship type, corresponds to field in a table  Relationships- link between entities  Strong entity- exists independently of other entity types and has can identifier  Weak entity- dependent on strong identify, may have partial identifier  **Normalization**  1st Normal Form – remove multivalued attributes  2nd Normal Form – in 1NF plus remove partial dependencies    3rd Normal Form – in 2NF plus remove transitive dependences | |
|  |  |
| **DCL – Data Control Language – Commands to grant rights to roles, users and groups on objects**   |  |  | | --- | --- | | Action/Command | Example | | GRANT | --grants select privileges to data604\_user on HR.Employees  GRANT SELECT ON HR.Employees TO data604\_user; | | REVOKE | --Removes select privileges to data604\_user on HR.Employees  REVOKE SELECT ON HR.Employees TO data604\_user; | | |
| **DDL – Data Definition Language- Commands to create and alter objects in database**   |  |  | | --- | --- | | Command | Example | | CREATE DATABASE | CREATE DATABASE TSQLV4; | | DROP DATABASE | DROP DATABASE TSQLV4 | | BACKUP DATABASE | BACKUP DATABASE TSQLV4  To DISK = 'D:\MSSQL\MSSQL14.MSSQLSERVER\MSSQL\Backup\Hw1tsql.bak' | | RESTORE DATABASE | RESTORE DATABASE [TSQLV4] FROM DISK = N'D:\MSSQL\MSSQL14.MSSQLSERVER\MSSQL\Backup\Hw1tsql.bak' WITH FILE = 3, NOUNLOAD, STATS = 5 | | CREATE TABLE | CREATE TABLE HR.Employee  (  employeeid INT NOT NULL IDENTITY,  lastname NVARCHAR(20) NOT NULL,  firstname NVARCHAR(10) NOT NULL,  title NVARCHAR(30) NOT NULL,  birthdate DATE NOT NULL,  hiredate DATE NOT NULL,  CONSTRAINT PK\_Employee PRIMARY KEY(empid)); | | ALTER TABLE | --adds two columns 'endDate and retireDate' to Employees table  ALTER TABLE [HR].[Employees] ADD endDate Date NULL, retireDate Date NULL ; | | CREATE VIEW | CREATE VIEW [Sales].[SalesStaff]  AS  SELECT lastname, firstname, title, hiredate  FROM HR.Employees  WHERE title like 'Sales%' | | CREATE STORED PROCEDURE | CREATE PROC pSalesStaff  @lastname VARCHAR(25)  as  -- exec pSalesStaff @lastname = 'Lew'  SELECT lastname, firstname, title, hiredate  FROM HR.Employees  WHERE title like 'Sales%' and lastname = @lastname | | |
| **DML- Data Manipulation Language – Comments to retrieve and modify data in tables**   |  |  | | --- | --- | | Action/Command | Syntax | | SELECT  Using Declared variables in a SELECT | SELECT \* FROM [HR].[Employees] WHERE YEAR(birthdate) > 1980 AND LEFT(firstname,1) = 'P'  --declare variable and assign value within SELECT Statement  DECLARE @ordid int  SELECT @ordid = min(orderid) FROM [Sales].[Orders]  PRINT @ordid | | INNER JOIN | /\*INNER JOIN to self, rows returned must have a match in both tables. This query returns employees if they have a manager \*/  SELECT e1.empid, e1.lastname, e1.firstname, e2.empid as mgrid, e2.lastname as mgrname, e2.firstname as mgrfname  FROM [HR].[Employees] e1 INNER JOIN [HR].[Employees] e2  on e1.mgrid = e2.empid | | LEFT OUTER JOIN | --left outer join, rows have an entry in the left table but may or may not have an entry in the right table. This query returns all employees and they may or may not have a manager  select e1.empid, e1.lastname, e1.firstname, e2.empid as mgrid, e2.lastname as mgrname, e2.firstname as mgrfname  from [HR].[Employees] e1 left outer join [HR].[Employees] e2  on e1.mgrid = e2.empid | | RIGHT OUTER JOIN | /RIGHT outer join returns all entries in the right table but may or may not have entry in left table. This query returns all the managers and the employees they manage\*/  select e2.empid as mgrid, e2.lastname as mgrname, e2.firstname as mgrfname, e1.empid, e1.lastname, e1.firstname  from [HR].[Employees] e1 RIGHT outer join [HR].[Employees] e2  on e1.mgrid = e2.empid | | SELECT…INTO | --SELECT INTO ...creates new temp table from select statement  SELECT empID, lastname, firstname INTO ##myTempTable  from HR.Employees  where Year(birthdate) > 1980 | | INSERT | INSERT INTO Sales.OrderDetails(orderid, productid, unitprice, qty, discount) VALUES(11036, 13, 6.00, 7, 0); | | DELETE | --Deletes row in temp table that has lastname of ‘Suurs’  DELETE  FROM ##myTempTable  WHERE lastname = 'Suurs' | | TRUNCATE | --Deletes all the data in the table without logging it  TRUNCATE TABLE ##myTempTable | | UNION | SELECT e.lastname + ', ' + e.firstname as fullname  FROM HR.Employees e  UNION  SELECT contactname as fullname FROM Sales.Customers | | |
| **Subqueries**   |  |  | | --- | --- | | Action/Command | Example | | Inline Subquery | select e.empid, e.lastname, 'lastorderdate' = (select max(orderdate) from Sales.Orders o where e.empid = o.empid)  from HR.Employees e | | Derived Table | --retrieves list of customers and last order date  SELECT c.custid, c.companyname, c.city, x.LastOrderDt  FROM  (SELECT MAX(orderdate) as LastOrderDt, custid  FROM Sales.Orders  GROUP BY custid) as x inner join Sales.Customers c on x.custid = c.custid | | |
| **Common Operators/Functions**   |  |  | | --- | --- | | Operator | Info/Example | | MIN()  MAX() and AVG() are similar | Returns minimum most value:  SELECT MIN(unitprice) as minPrice  FROM Sales.OrderDetails  Can add ‘Group By’ clause to return value for each group:  SELECT productID, MIN(unitprice) as minPrice  FROM Sales.OrderDetails  GROUP BY productID | | DISTINCT | Removes redundant rows: SELECT DISTINCT title FROM [HR].[Employees] | | IF | DECLARE @custid int = 0  IF @custid = 0  BEGIN  PRINT 'will not execute'  END  ELSE  BEGIN  PRINT 'will execute'  SELECT \* FROM sales.salestaff  END | | CASE  Or IIF | SELECT lastname, firstname, title,  'new title' = CASE  WHEN title = 'CEO' THEN 'top emp'  WHEN title like 'Sales%' THEN 'Sales Staff'  ELSE 'General Staff' END,  IIF(title = 'CEO','top emp',IIF(title like 'Sales%','Sales Staff','General Staff')) as anotherWay  FROM [HR].[Employees] | | WHILE | DECLARE @ordid int, @ordercount int  SELECT @ordid = MIN(orderid) FROM [Sales].[Orders]  SELECT @ordercount = 0  WHILE not @ordid is null  BEGIN  SELECT @ordercount = @ordercount + 1  SELECT @ordid = MIN(orderid) FROM[Sales].[Orders]  WHERE orderid > @ordid  End  PRINT 'Total orders are ' + CAST(@ordercount as varchar) | | REPLACE() | Replaces value found in a string with another value SELECT [title], REPLACE(title, 'Sales','Customer') as newTitle  FROM [TSQLV4].[HR].[Employees] | | |