Most Used Queries

- SELECT **DISTINCT** fieldname FROM tablename WHERE condition
- SELECT **TOP *N*** fieldname FROM tablename WHERE condition
- SELECT **COUNT** (fieldnames), **SUM** (Purhases) FROM tablename WHERE condition
- SELECT **MAX** (numberfield) , **MIN** (numberfield) FROM tablename WHERE condition

Use nested SELECT statement as a fieldname

- SELECT fieldname **AS** showfield, **SELECT MiddleName FROM tablename AS MidName**, another field WHERE condition

Data Selection on WHERE using (AND, OR, BETWEEN conditionals)

- …**WHERE** date = '20091018' **AND** date = '20101210'
- …**WHERE** date = '20091018' **OR** (customerID = 224423 **AND** date ='20090507')
- …**WHERE** date **BETWEEN** '20091018' **AND** '20101210'
- … **WHERE** Name **LIKE** | **NOT LIKE** '%ema*'
- … **WHERE** SomeID **IS NULL** | **IS NOT NULL**

GROUP BY / ORDER BY / HAVING

- **SELECT** * FROM Employees **ORDER BY** 2, 3
- **SELECT** CustomerID, Invoices **FROM** Sales **GROUP BY** CustomerID

 # Group by after WHERE clause:

SELECT * FROM Table **WHERE** someID > 200 **GROUP BY** zipCode

#Using HAVING clause

SELECT CustomerID, Count(*) **AS** Orders FROM Table
GROUP BY CustomerID
HAVING Count(*) >=4

SELECT shipdate, salesorderdate, customerid
CASE WHEN Freight < 25 THEN 'lite'
WHEN Freight **BETWEEN** 25 **AND** 100 THEN 'normal'
WHEN Freight > 101 **THEN** 'heavy'
ELSE 'None'
END FrieightType - - This is a new column
FROM schemanamespace.tablename
WHERE *condition*

 # INSERT Methods ⬇

- **INSERT INTO** table **VALUES** (val1, val2, val3)
- **INSERT INTO** Table(fieldname) **VALUES** (value1)

 # Copy Table Data to a new table

- **SELECT** * **INTO** NewTable **FROM** ExistingTable

 # Copy Data in a new column

- **UPDATE** Table **SET** newcolumn = originalcolumn
- **BULK INSERT** database.schemanamespace.tablename **FROM** 'c:\filename.txt'
- **BCP** AdventureWorks.Production.ProductName **in** "Products.txt" **–T -C**

Sub-Queries with ALL, ANY, IN and EXISTS

 # IN Conditional

SELECT CustomerID, FirstName, LastName FROM Customers
WHERE CustomerID **IN**
(SELECT CustomerID **FROM** Sales WHERE SupplyID **LIKE** 'C*')
you can chain more IN statements from here
IN (SELECT …)
SELECT CustomerID, MAX(DateSold)

 # EXISTS Conditional

SELECT * FROM Customers
WHERE **EXISTS**
(SELECT * FROM Sales WHERE CustomerID = SalesID)

 # ANY *(can use any comparison operator <, > , <>, = ANY)*

SELECT CustomerID, FirstName, LastName FROM Customers
WHERE City **= ANY** (SELECT City FROM Customers WHERE State = 'FL')

 # ALL *(can use any comparison operator <, > , <>, = ALL)*

SELECT CustomerID, FirstName, LastName FROM Customers
WHERE ZipCode **> ALL** (SELECT ZipCode FROM Customers WHERE State = 'CA')

INNER JOINS – equi-joins

Remember **FROM** tbl **JOIN** tbl **ON** relation
SELECT SAL_SOD.SalesOrderID, SAL_SOD.SalesOrderDetailID
PRO_P.ProductID, PRO_P.ProductName,
FROM Sales.SalesOrderDetail SAL_SOD **INNER JOIN** Production.Product PROP_P
ON SAL_SOD.ProductID=PRO_P.ProductID
WHERE SAL_SOD.SalesOrderID > 46559

 # Syntax starting from the FROM clause

FROM Table2 (**LEFT** | **RIGHT**){ **INNER** | **OUTER**} **JOIN** Table2 **ON** Table1Relation = Table2Relation

CROSS JOINS = Cartesian Product
SELECT * FROM Table1 CROSS JOIN Table2

Table Management

- **CREATE TABLE (** fieldname NVARCHAR (10) **NOT NULL PRIMARY KEY,**
  FOREIGN KEY (socialsecuritynumber**) REFERENCES** Table2 **(** socialsecuritynumber**))**

- **ALTER TABLE** Tablename **ADD** *ColumnName DataType*
- **DROP TABLE** Tablename

Remove column requires ALTER TABLE preceding DROP COLUMN

- **ALTER TABLE** Tablename **DROP COLUMN** ColumnName

T-SQL SYNTAX BLOCK

- Defining and using variables (**DECLARE**, **SET**)
- Controlling Flow (**IF**, **ELSE**, **RETURN**)
- Using Loops (**WHILE** , **BREAK**, **CONTINUE**)
- Defining a block (**BEGIN**, **END**)

***# SCALAR UDF** – must return scalar type (string, number, or date value NOT a table or view)*

```
CREATE FUNCTION ownername.functionname (@varname  type )
RETURNS  returntype
AS  BEGIN
        DECLARE @amount MONEY
        SET @amount = 0
        SELECT  @amount = AVG(TotalDue)
        FROM       Sales.SalesOrderHeader
        WHERE (CustomerID  = @CustomerID)
        RETURN ISNULL (@Amount, 0)
END
```

#Create an Index

- **CREATE INDEX** IndexName **ON** Tablename (Column1, Column2)

#Create a View

- **CREATE VIEW** Viewname **AS SELECT** *statement…*

 # Create and Execute an SP

- **CREATE PROCEDURE** CountPurchases **AS** *SELECT statement…*
- **EXEC** CountPurchases

```
CREATE PROCEDURE SalesStats
@spid varchar(5)
AS SELECT statement
```

```
CREATE TRIGGER Triggername ON Tablename  FOR INSERT, UPDATE , DELETE AS  SQL Operation
```

 # Creating a TRANSACTION

```
BEGIN TRANSACTION
INSERT INTO Customers  VALUES ( 11, 'Hot' , '1222 Street') - - do some SQL operation
SAVE TRANSACTION TransactionName
IF @@ERROR <> 0
        BEGIN
        PRINT "An Error occurred here" - - error handler here
        ROLLBACK TRANSACTION TransactionName
        END
COMMIT TRANSACTION
```

***# TABLE VALUE UDF** – returns a table instead*

```
CREATE FUNCTION dbo.GetInterest (@NumPeriods int, @PercentInterst money)
RETURNS ( Num int, I money)
AS BEGIN
        DECLARE @N = 0
        SET @N = 0
        DECLARE @ITot money
        SET @ITot = 1
        WHILE @N < @NumPeriods
                BEGIN
                SET @N = @N + 1
                SET @ ITot = @ITot * (1 + (@PercentInterest / 100 ))
                INSERT INTO @InterestTable VALUES (@N, @ITot )
                END
                RETURN
END
```

Using IF ELSE

```
DECLARE @compareprice money, @cost money
EXECUTE Production.uspGetList '%Bikes%', 700,
  @compareprice OUT,
  @cost OUTPUT
IF @cost <= @compareprice
BEGIN
  PRINT 'These products can be purchased for less than
END
ELSE
  PRINT 'The prices for all products in this category exceed
  $'+ RTRIM(CAST(@compareprice AS varchar(20)))+'.'
```

Create a CURSOR

```
DECLARE CursorName CURSOR FOR SELECT statement OPEN CursorName FETCH NEXT FROM CursorName
WHILE @@FETCH STATUS = 0
BEGIN FETCH NEXT FROM CursorName  - - include a SQL operation
END
CLOSE CursorName
DEALLOCATE CursorName
```

GLOBAL VARIABLES

```
@@VERSION
@@ERROR
@@ROWCOUNT
@@TRANCOUNT  #nesting level of a txn
@@IDENTITY
```

- **F5 (**Execute)
- **CTRL + F5** (Parse)
- **CTRL + SHIFT + Q** (Query Designer)
- **CTRL + L** (Estimated Execution Plan)
- **SP_HELP** (Show Tables and Details)
- **CTRL + N** (New Query)

Sybase:

- SELECT * FROM sysobjects WHERE type='U'