--This article will discuss and illustrate how to use aggregate functions such as:

--SUM – provides the total non - null value

--AVG – provides the average non - null value

--MIN – provides the lowest non - null value

--MAX – provides the maximum non - null value

--COUNT – provides the number non - null value

--COUNT (\*) – provides the number of rows

--Note: the SUM and AVG function must result in a numeric value, while the MIN, MAX, and COUNT can result in numeric, string or date.

USE AdventureWorks2012

GO

Create Table Books

(BooksID int Identity (1,1) Not Null Primary Key,

BookTitle varchar (50) Null,

BookAuthor varchar (50) Null,

BookQuantity int Null,

SoldDate Datetime Null)

SELECT \* FROM Books

--Insert data into table

Insert Into Books

Values

('The Great Gatsby 2','F Scott Fitzgerald',32325,'02-10/11'),

('Pride and Prejudice','Jane Austen',32,'03-10/15'),

('The Lord of the Rings','JRR Tolkien',555,'03-10/15'),

('Jane Eyre','Charlotte Bronte',3454,'03-10/15'),

('Harry Potter series','JK Rowling',5434,'02-10/15'),

('To Kill a Mockingbird','Harper Lee',866,'02-10/15'),

('Wuthering Heights','Emily Bronte',45646,'02-10/15'),

('Nineteen Eighty Four','George Orwell',34523,'01-10/15'),

('His Dark Materials','Philip Pullman',45453,'01-10/15'),

('Great Expectations','Charles Dickens',23432,'01-10/15'),

('Little Women','Louisa M Alcott',34234,'03-10/14'),

('Tess of the D’,Urbervilles', 'Thomas Hardy',234234,'03-10/14'),

('Catch 22','Joseph Heller',2343,'03-10/13'),

('Rebecca','Daphne Du Maurier',24342,'03-10/13'),

('The Hobbit','JRR Tolkien',342343,'03-10/13'),

('Birdsong','Sebastian Faulk',23432,'03-10/13'),

('Catcher in the Rye','JD Salinger',5756,'02-10/12'),

('The Time Traveler’s Wife','Audrey Niffenegger',7564,'02-10/11'),

('Middlemarch','George Eliot',909090,'02-10/11'),

('Gone With The Wind','Margaret Mitchell',8844,'02-10/11'),

('The Great Gatsby','F Scott Fitzgerald',90494,'02-10/11')

SELECT \* FROM Books

--How many total books were sold? Use SUM command

SELECT SUM (BookQuantity) AS TotalBooksSold

FROM books

SELECT \*

FROM Books

ORDER BY SoldDate

--How many books where sold in year 2011?

SELECT SUM (BookQuantity) AS ProductTotal

FROM Books

WHERE SoldDate = '2011-02-10 00:00:00.000'

--What was the total number of books sold by F Scott Fitzgerald?

Select BookAuthor, SUM(BookQuantity) AS TotalBooksSold--<< Use an alias for column title

FROM Books

WHERE BookAuthor = 'F Scott Fitzgerald'

GROUP BY BookAuthor

USE AdventureWorks2012

GO

CREATE TABLE EmpSalary

(EmpSalaryID int Identity (1,1) Not Null Primary Key,

Fname varchar (20) Null,

Lname varchar (20) Null,

Salary Money Null,

Sales Money Null,

Commission varchar (10) Null)

Select \* from EmpSalary

Insert Into EmpSalary

values

('Tom','Smith','35000',453000,'2'),

('Stan','Brimes','34055',7686,'10'),

('Roger','Fuller','23045',34834,'3'),

('Ralph','Knowes','76034',96675,'7'),

('Andy','Mattews','86076',21193,'10')

--What is the sum of sales and commission or each rep? (Using SUM with multiple columns)

Select Fname,Lname,Salary,Sales, SUM(Sales \* Commission) AS TotalBonus

from EmpSalary

group by Fname,Lname,Salary,Sales

--Using the AVG function. What is the average sales?

Select AVG(Sales) From EmpSalary

--Who has the highest salary?

Select fname,lname, MAX(Salary) AS HighestSalary

From EmpSalary

GROUP BY fname,lname

ORDER BY HighestSalary DESC --<< Note. Using the alias as column name in the Order By clause

--Who has the lowest salary?

Select fname,lname, MIN(Salary) AS LowestSalary

From EmpSalary

GROUP BY fname,lname

ORDER BY LowestSalary

--Combining aggregates in a single query

Select SUM (Sales) AS TotalSales,

AVG (Sales) AS AveSales,

MAX (Sales) AS MAXSales,

MIN (Sales) AS MINSales

FROM EmpSalary

--Use of count

SELECT COUNT([SalesOrderID]) AS TotalSalesOrderIDs

,[SalesOrderDetailID]

,[CarrierTrackingNumber]

,[OrderQty]

,[ProductID]

,[UnitPrice]

FROM [AdventureWorks2012].[Sales].[SalesOrderDetail]

GROUP BY

[SalesOrderDetailID]

,[CarrierTrackingNumber]

,[OrderQty]

,[ProductID]

,[UnitPrice]

SELECT AVG (InvoiceTotal) AS InvoiceAve

FROM Invoices

SELECT MIN (InvoiceTotal) AS InvoiceMin

FROM Invoices

SELECT MAX (InvoiceTotal) AS InvoiceMax

FROM Invoices

SELECT COUNT (InvoiceTotal) AS TotalNumberOfinvoices

FROM Invoices

SELECT COUNT (\*) AS TotalNumberOfinvoices

FROM Invoices

select Vendorid

from Invoices

group by VendorID

order by VendorID

SELECT COUNT (DISTINCT VendorID)

FROM Invoices

--multiple aggregates in a single select statement

SELECT

SUM (InvoiceTotal) AS [InvoiceTotal],

AVG (InvoiceTotal) AS InvoiceaAve,

MIN (InvoiceTotal) AS InvoiceMin,

MAX (InvoiceTotal) AS InvoiceMax,

COUNT (InvoiceTotal) AS TotalNumberOfinvoices

FROM Invoices

--Using max and min for non numeric values

SELECT MAX (VendorName) AS [VendorName], MIN (VendorName) AS [VendorName]

FROM Vendors

--Using GROUP BY and HAVING commands with aggregates.

--Give the average invoice total for each vendor greater that 2000.

SELECT Vendorid, AVG(InvoiceTotal) AS InvoiceTotalAve, COUNT(\*) as NumberOfInvoices

FROM Invoices

GROUP BY Vendorid

HAVING AVG(InvoiceTotal) > 2000

ORDER BY InvoiceTotalAve --using the alias

--which vendor has the highest invoice

select max(Invoicetotal) as HighestTotal, VendorName

from Invoices

join Vendors

on Invoices.InvoiceID = Vendors.VendorID

group by VendorName

order by HighestTotal desc

SELECT \* FROM Invoices

WHERE VendorID = 115

--give the average total by city and state

SELECT COUNT(\*) AS TotalNumberOfInvoices, AVG(InvoiceTotal) AS InvoiceTotalAve, VendorCity, VendorState

From Invoices JOIN Vendors

ON Invoices.Vendorid = Vendors.Vendorid

GROUP BY VendorCity, VendorState

--USING HAVING TO FILTER GROUPS

SELECT COUNT(\*) AS TotalNumberOfInvoices, SUM (InvoiceTotal) AS [InvoiceTotal],AVG(InvoiceTotal) AS InvoiceTotalAve, VendorCity, VendorState

From Invoices JOIN Vendors

ON Invoices.Vendorid = Vendors.Vendorid

WHERE VendorState = 'CA'

GROUP BY VendorCity, VendorState

HAVING COUNT(\*) > 3

--give the balance due

SELECT Invoicetotal,PaymentTotal,CreditTotal,(Invoicetotal - PaymentTotal - CreditTotal) AS [Balance Due], InvoiceNumber,InvoiceDueDate,VendorName

FROM Invoices

join Vendors

ON invoices.InvoiceID = Vendors.VendorID

WHERE (Invoicetotal - PaymentTotal - CreditTotal) > 0

--how many invoices are there for each vendor?

SELECT VendorID, count(\*) AS TotalVendorsInvoices

FROM Invoices

GROUP BY VendorID

HAVING count(\*) >= 8

ORDER BY VendorID

DESC

-------------------------------------End Session