## SELECT clause

SELECT 1+1 AS Result, 3\*3 As Result2, 4\*4 as [Result 3]

## From clause

SELECT [name] AS NameOfObject, [object\_id], [type\_desc]

FROM sys.objects

## WHERE clause

SELECT \*

FROM sys.objects

WHERE object\_id >= 5 AND object\_id<=9

SELECT \*

FROM sys.objects

WHERE object\_id between 5 AND 9

SELECT \*

FROM sys.objects

WHERE object\_id = 5 OR object\_id = 9

SELECT \*

FROM sys.objects

WHERE type\_desc = 'SYSTEM\_TABLE'

SELECT \*

FROM sys.objects

WHERE [name] LIKE 'sysr%'

SELECT \*

FROM sys.objects

WHERE [name] LIKE '%a%'

-- % 0-infinite characters

-- \_ 1 character

SELECT \*

FROM sys.objects

WHERE create\_date < '2019-01-01'

## GROUP BY

SELECT schema\_id, name, count(\*) as NumberOfRows, sum(schema\_id)

FROM sys.objects

GROUP BY schema\_id, name

SELECT distinct schema\_id

FROM sys.objects

SELECT TOP (10) \*

FROM sys.objects

## HAVING clause

SELECT name, count(\*) as NumberOfRows

FROM sys.objects

WHERE name LIKE 'f%'

GROUP BY name

HAVING count(\*)>1

## ORDER BY clause

SELECT name

FROM sys.objects

ORDER BY name DESC

SELECT name, count(\*) as NumberOfRows

FROM sys.objects

WHERE name LIKE 'f%'

GROUP BY name

--HAVING count(\*)>1

ORDER BY NumberOfRows ASC, name DESC

## Numeric types

SELECT 5/2 AS Result

## Date/time types

DECLARE @variable time(7) = '2024-03-02 12:34:56.1234567'

SELECT @variable as Answer

## Character types

DECLARE @variable nvarchar(max) = N'My name is PhillipⱩ'

SELECT @variable as Answer

## INNER JOIN

select E.EmployeeNumber, EmployeeFirstName, Amount, DateOfTransaction

from [dbo].[tblEmployee] AS E

JOIN [dbo].[tblTransaction] AS T

ON E.EmployeeNumber = T.EmployeeNumber

## Other JOINs

select E.EmployeeNumber, EmployeeFirstName, Amount, DateOfTransaction

from [dbo].[tblEmployee] AS E

LEFT JOIN [dbo].[tblTransaction] AS T

ON E.EmployeeNumber = T.EmployeeNumber

## IS NULL – Missing Data

select E.EmployeeNumber

from [dbo].[tblEmployee] AS E

LEFT JOIN [dbo].[tblTransaction] AS T

ON E.EmployeeNumber = T.EmployeeNumber

WHERE T.EmployeeNumber IS NULL

## Self Join

SELECT E1.\*, E2.EmployeeFirstName AS ManagerFirstName, E2.EmployeeLastName AS ManagerLastName

FROM [dbo].[tblEmployee] AS E1

LEFT JOIN [dbo].[tblEmployee] AS E2

ON E1.Manager = E2.EmployeeNumber

## Add/Remove Columns

CREATE TABLE tblTest

(EmployeeNumber tinyint NOT NULL)

ALTER TABLE tblTest

ALTER COLUMN EmployeeNumber smallint NOT NULL

ALTER TABLE tblTest

ADD [Transaction] smallmoney NOT NULL

ALTER TABLE tblTest

DROP COLUMN [Transaction]

DROP TABLE tblTest

TRUNCATE TABLE tblTest

## PRIMARY KEY Constraints

select \* from [dbo].[tblEmployee]

where EmployeeNumber = 4

ALTER TABLE tblEmployee

ADD CONSTRAINT PK\_tblEmployee\_EmployeeNumber PRIMARY KEY ([EmployeeNumber])

ALTER TABLE tblEmployee

DROP CONSTRAINT PK\_tblEmployee\_EmployeeNumber

CREATE TABLE tblTest

(EmployeeNumber int not null,

CONSTRAINT PK\_tblTest\_EmployeeNumber PRIMARY KEY ([EmployeeNumber])

)

## Composite PRIMARY KEYs

ALTER TABLE [dbo].[tblTransaction]

ADD CONSTRAINT PK\_tblTransaction\_EmployeeNumber\_DateOfTransaction

PRIMARY KEY (EmployeeNumber, DateOfTransaction)

ALTER TABLE [dbo].[tblTransaction]

DROP CONSTRAINT PK\_tblTransaction\_EmployeeNumber\_DateOfTransaction

ALTER TABLE [dbo].[tblTransaction]

ADD IDTransaction int IDENTITY(1,1) NOT NULL

ALTER TABLE [dbo].[tblTransaction]

ADD CONSTRAINT PK\_tblTransaction\_IDTransaction PRIMARY KEY (IDTransaction)

## FOREIGN KEYs

ALTER TABLE [dbo].[tblTransaction] WITH NOCHECK

ADD CONSTRAINT FK\_tblTransaction\_EmployeeNumber

FOREIGN KEY ([EmployeeNumber])

REFERENCES [dbo].[tblEmployee]([EmployeeNumber])

## Understand the Relationship between PRIMARY KEYs and FOREIGN KEYs

ALTER TABLE [dbo].[tblTransaction] WITH NOCHECK

ADD CONSTRAINT FK\_tblTransaction\_EmployeeNumber

FOREIGN KEY ([EmployeeNumber])

REFERENCES [dbo].[tblEmployee]([EmployeeNumber])

ON DELETE CASCADE

ON UPDATE CASCADE

--NO ACTION (Error)

--CASCADE

--SET NULL

--SET DEFAULT

ALTER TABLE [dbo].[tblTransaction]

DROP CONSTRAINT FK\_tblTransaction\_EmployeeNumber

SELECT \* FROM [dbo].[tblEmployee] WHERE [EmployeeNumber] IN (5,13)

SELECT \* FROM [dbo].[tblTransaction] WHERE [EmployeeNumber] IN (5,13) OR IDTransaction IN (10, 11, 12)

BEGIN TRAN

DELETE FROM [dbo].[tblEmployee]

WHERE EmployeeNumber = 5

SELECT \* FROM [dbo].[tblEmployee] WHERE [EmployeeNumber] IN (5,13)

SELECT \* FROM [dbo].[tblTransaction] WHERE [EmployeeNumber] IN (5,13) OR IDTransaction IN (10, 11, 12)

ROLLBACK TRAN

## Other Constraints

ALTER TABLE [dbo].[tblTransaction]

ADD CONSTRAINT DF\_tblTransaction\_Amount

DEFAULT 0 FOR [Amount]

CREATE TABLE tblTest2

(EmployeeNumber int DEFAULT 0)

DROP TABLE tblTest2

CREATE TABLE tblTest2

(EmployeeNumber int CONSTRAINT DF\_tblTest2\_Employee DEFAULT 0)

ALTER TABLE [dbo].[tblEmployee]

ADD CONSTRAINT UQ\_tblEmployee\_EmployeeGovernmentID

UNIQUE ([EmployeeGovernmentID])

ALTER TABLE [dbo].[tblTransaction]

ADD CONSTRAINT CK\_tblTransaction\_Amount

CHECK ((Amount>=-1000 and Amount<=1000) or EmployeeNumber = 2)

## Data Manipulation Language (DML) Introduction

Data Manipulation Language (DML)

SELECT

INSERT

UPDATE

DELETE

Data Definition Language (DDL)

CREATE TABLE

ALTER TABLE

DROP TABLE

## How data is inserted into tables

Transactions

BEGIN TRAN

COMMIT TRAN

ROLLBACK TRAN

INSERT --Statement1

insert --Statement2

--Is the same as

BEGIN TRAN

INSERT --Statement1

COMMIT TRAN

begin tran

insert --Statement2

COMMIT TRAN

implicit transactions

ACID

Atomicity

Consistency

Isolation

Durability

## How to use INSERT statements – VALUES

INSERT INTO [dbo].[tblEmployee]

([EmployeeNumber],[EmployeeFirstName],[EmployeeMiddleName],[EmployeeLastName],[EmployeeGovernmentID]

,[DateOfBirth],[Department],[Manager])

VALUES (2, 'Jane', NULL, 'Zwilling', 'AB123456G ', '1994-12-30', 'Customer Relations', NULL),

(3, 'Carolyn', 'Andrea', 'Zimmerman', 'AB234578H ', '1985-05-29', 'Commercial', '2'),

(4, 'Jane', NULL, 'Zabokritski', 'LU778728T ', '1987-12-07', 'Commercial', '2'),

(5, 'Ken', 'J', 'Yukish', 'PO201903O ', '1979-12-25', 'HR', '2'),

(6, 'Terri', 'Lee', 'Yu', 'ZH206496W ', '1996-11-11', 'Customer Relations', '2'),

(7, 'Roberto', NULL, 'Young', 'EH793082D ', '1977-04-02', 'Customer Relations', '3'),

(8, 'Rob', NULL, 'Yalovsky', 'WF039886Z ', '1991-08-29', 'Litigation', '4'),

(9, 'Gail', 'A', 'Wu', 'SR883921U ', '1982-02-16', 'HR', '6'),

(10, 'Jossef', 'H', 'Wright', 'FU781952O ', '1990-07-30', 'Commercial', '5'),

(11, 'Dylan', 'A', 'Word', 'SU416128W ', '1999-11-26', 'Customer Relations', '5');

## How to use INSERT statements – from tables

INSERT INTO [dbo].[tblEmployee2]

SELECT \*

FROM [dbo].[tblEmployee]

WHERE EmployeeNumber in (2, 3, 6)

select \* from [dbo].[tblEmployee2]

SELECT \*

INTO [dbo].[tblEmployee3]

FROM [dbo].[tblEmployee]

WHERE EmployeeNumber in (2, 3, 6)

## Understand how data is updated in a database

BEGIN TRAN

UPDATE

COMMIT TRAN

ROLLBACK TRAN

lock row

page

table

## How to write the updated data to the database by using UPDATE statement

CREATE TABLE tblEmployeeChanges

(NewEmployeeNumber int,

NewEmployeeGovernmentID char(10))

SELECT \* FROM tblEmployee

select \*

from [dbo].[tblEmployeeChanges]

BEGIN TRAN

UPDATE [dbo].[tblEmployee]

SET [EmployeeGovernmentID] = [NewEmployeeGovernmentID]

FROM [dbo].[tblEmployeeChanges]

WHERE EmployeeNumber = [NewEmployeeNumber]

SELECT \* FROM tblEmployee

ROLLBACK TRAN

BEGIN TRAN

UPDATE [dbo].[tblEmployee]

SET [EmployeeGovernmentID] = B.[EmployeeGovernmentID]

FROM [dbo].[tblEmployee] AS A

JOIN [dbo].[tblEmployeeChanges] AS B

ON A.EmployeeNumber = B.[EmployeeNumber]

SELECT \* FROM tblEmployee

ROLLBACK TRAN

## DELETE rows

BEGIN TRAN

DELETE

FROM [dbo].[tblEmployee]

WHERE EmployeeNumber = 9

SELECT \* FROM [dbo].[tblEmployee]

ROLLBACK TRAN

BEGIN TRAN

SELECT T.\*

INTO dbo.tblTransactionToBeDeleted

FROM [dbo].[tblTransaction] AS T

LEFT JOIN [dbo].[tblEmployee] AS E

ON T.EmployeeNumber = E.EmployeeNumber

WHERE E.EmployeeNumber IS NULL

DELETE FROM T

FROM [dbo].[tblTransaction] AS T

LEFT JOIN [dbo].[tblEmployee] AS E

ON T.EmployeeNumber = E.EmployeeNumber

WHERE E.EmployeeNumber IS NULL

SELECT \* FROM [dbo].[tblTransaction]

ROLLBACK TRAN

## Data Definition Language (DDL)

Data Manipulation Language (DML)

SELECT

INSERT

UPDATE

DELETE

Data Definition Language (DDL)

CREATE

ALTER

DROP

## When to use views

-- encapsulation

-- security

-- simplicity

-- column name simplification

-- multi-part queries

-- source data changes

## Creating views

CREATE VIEW MyViewSQL AS

SELECT \*

FROM [dbo].[tblEmployee]

WHERE EmployeeNumber < 10

## Updating rows using views

BEGIN TRAN

UPDATE [dbo].[MyViewSQL]

SET EmployeeGovernmentID = 'ABC123'

Where EmployeeNumber = 6

SELECT \* FROM [dbo].[MyViewSQL]

SELECT \* FROM [dbo].[tblEmployee]

ROLLBACK TRAN

CREATE VIEW EmployeeTransaction as

SELECT E.\*, T.Amount, T.DateOfTransaction

FROM tblEmployee as E

LEFT JOIN tblTransaction as T

ON E.EmployeeNumber = T.EmployeeNumber

BEGIN TRAN

insert into [dbo].[EmployeeTransaction]

([EmployeeNumber]

,[EmployeeFirstName]

,[EmployeeMiddleName]

,[EmployeeLastName]

,[EmployeeGovernmentID]

,[DateOfBirth]

,[Department]

,[Manager]

,[Amount]

,[DateOfTransaction])

values (12, 'Dylan', 'A', 'Word', 'SU416128X ', '1999-11-26', 'Customer Relations', '5', 1.23, '2025-01-01')

select \* from [dbo].[tblEmployee]

--DELETE

--from EmployeeTransaction

--WHERE EmployeeNumber = 10

ROLLBACK TRAN

## With Check Option

DROP VIEW EmployeeTransaction

GO

CREATE VIEW EmployeeTransaction as

SELECT E.\*, T.Amount, T.DateOfTransaction

FROM tblEmployee as E

LEFT JOIN tblTransaction as T

ON E.EmployeeNumber = T.EmployeeNumber

WHERE E.EmployeeNumber < 10

WITH CHECK OPTION

GO

BEGIN TRAN

insert into [dbo].[EmployeeTransaction]

([EmployeeNumber]

,[EmployeeFirstName]

,[EmployeeMiddleName]

,[EmployeeLastName]

,[EmployeeGovernmentID]

,[DateOfBirth]

,[Department]

,[Manager])

values (1, 'Dylan', 'A', 'Word', 'SU416128X ', '1999-11-26', 'Customer Relations', '5'),

(12, 'Dylan', 'A', 'Word', 'SU416128X ', '1999-11-26', 'Customer Relations', '5')

select \* from EmployeeTransaction

SELECT \* FROM [dbo].[tblEmployee]

ROLLBACK TRAN

BEGIN TRAN

UPDATE EmployeeTransaction

Set EmployeeNumber = EmployeeNumber + 10

select \* from EmployeeTransaction

SELECT \* FROM [dbo].[tblEmployee]

ROLLBACK TRAN

## WITH SCHEMABINDING

DROP VIEW EmployeeTransaction

GO

CREATE VIEW EmployeeTransaction WITH SCHEMABINDING as

SELECT E.[EmployeeNumber]

,[EmployeeFirstName]

,[EmployeeMiddleName]

,[EmployeeLastName]

,[EmployeeGovernmentID]

,[Department]

,[Manager], T.Amount, T.DateOfTransaction

FROM [dbo].[tblEmployee] as E

LEFT JOIN [dbo].[tblTransaction] as T

ON E.EmployeeNumber = T.EmployeeNumber

WHERE E.EmployeeNumber < 10

WITH CHECK OPTION

GO

ALTER TABLE [dbo].[tblEmployee]

DROP COLUMN [EmployeeMiddleName] --Doesn't work; it is protected by the view.

## Create Stored Procedures

ALTER PROC proc\_EmployeeTransaction (@EmployeeNumber INT) AS

SELECT \*

FROM EmployeeTransaction

WHERE EmployeeNumber = @EmployeeNumber

SELECT \*

FROM [dbo].[tblTransaction]

WHERE EmployeeNumber = @EmployeeNumber

EXEC proc\_EmployeeTransaction 2

## Create functions

-- scalar-valued user defined function

CREATE FUNCTION fn\_TransactionCount (@EmployeeNumber INT)

RETURNS INT

AS

BEGIN

DECLARE @NumOfTransaction int

SELECT @NumOfTransaction = COUNT(\*)

FROM [dbo].[tblTransaction]

WHERE EmployeeNumber = @EmployeeNumber

RETURN (@NumOfTransaction)

END

GO

select \*, dbo.fn\_TransactionCount(EmployeeNumber) as NumberOfTransactions

from [dbo].[tblEmployee]

## AFTER TRIGGER

insert into [dbo].[tblEmployee]

([EmployeeNumber]

,[EmployeeFirstName]

,[EmployeeMiddleName]

,[EmployeeLastName]

,[EmployeeGovernmentID]

,[DateOfBirth]

,[Department]

,[Manager])

values (1, 'Dylan', 'A', 'Word', 'SU416128X ', '1999-11-26', 'Customer Relations', '5'),

(12, 'Dylan', 'A', 'Word', 'SU416128X ', '1999-11-26', 'Customer Relations', '5')

CREATE TRIGGER trg\_tblEmployee\_After

ON [dbo].[tblEmployee]

AFTER INSERT, UPDATE, DELETE

AS

BEGIN

INSERT INTO [dbo].[tblEmployee2]

([EmployeeNumber],[EmployeeFirstName],[EmployeeMiddleName],[EmployeeLastName],[EmployeeGovernmentID]

,[DateOfBirth],[Department],[Manager], TypeOfChange, TimeOfChange)

SELECT [EmployeeNumber],[EmployeeFirstName],[EmployeeMiddleName],[EmployeeLastName],[EmployeeGovernmentID]

,[DateOfBirth],[Department],[Manager], 'Inserted', GETDATE()

FROM inserted

INSERT INTO [dbo].[tblEmployee2]

([EmployeeNumber],[EmployeeFirstName],[EmployeeMiddleName],[EmployeeLastName],[EmployeeGovernmentID]

,[DateOfBirth],[Department],[Manager], TypeOfChange, TimeOfChange)

SELECT [EmployeeNumber],[EmployeeFirstName],[EmployeeMiddleName],[EmployeeLastName],[EmployeeGovernmentID]

,[DateOfBirth],[Department],[Manager], 'Deleted', GETDATE()

FROM deleted

END

SELECT \* FROM [dbo].[tblEmployee2]

TRUNCATE TABLE [dbo].[tblEmployee2]

update [dbo].[tblEmployee]

Set EmployeeFirstName = 'J'

WHERE EmployeeNumber in (1, 12)

## INSTEAD OF TRIGGER

DROP VIEW EmployeeTransaction

GO

CREATE VIEW EmployeeTransaction as

SELECT E.\*, T.Amount, T.DateOfTransaction

FROM tblEmployee as E

LEFT JOIN tblTransaction as T

ON E.EmployeeNumber = T.EmployeeNumber

GO

insert into [dbo].[EmployeeTransaction]

([EmployeeNumber]

,[EmployeeFirstName]

,[EmployeeMiddleName]

,[EmployeeLastName]

,[EmployeeGovernmentID]

,[DateOfBirth]

,[Department]

,[Manager]

,[Amount]

,[DateOfTransaction])

values (12, 'Dylan', 'A', 'Word', 'SU416128X ', '1999-11-26', 'Customer Relations', '5', 1.23, '2025-01-01')

CREATE TRIGGER tr\_EmployeeTransaction ON [dbo].[EmployeeTransaction]

INSTEAD OF INSERT

AS

BEGIN

insert into [dbo].[tblEmployee]

([EmployeeNumber],[EmployeeFirstName],[EmployeeMiddleName],[EmployeeLastName],[EmployeeGovernmentID],[DateOfBirth]

,[Department],[Manager])

SELECT [EmployeeNumber],[EmployeeFirstName],[EmployeeMiddleName],[EmployeeLastName],[EmployeeGovernmentID],[DateOfBirth]

,[Department],[Manager]

FROM inserted

INSERT INTO [dbo].[tblTransaction]

([EmployeeNumber],[Amount],[DateOfTransaction])

SELECT [EmployeeNumber],[Amount],[DateOfTransaction]

FROM inserted

END

select \* from [dbo].[tblEmployee] WHERE EmployeeNumber in (1,12)

select \* from [dbo].[tblTransaction] WHERE EmployeeNumber in (1,12)

go

## Set Operators

select [EmployeeNumber],[EmployeeFirstName],[EmployeeMiddleName],[EmployeeLastName]

,[EmployeeGovernmentID],[DateOfBirth],[Department],[Manager]

from [dbo].[tblEmployeePart1]

UNION -- UNION ALL

select [EmployeeNumber] AS NumberOfEmployee,[EmployeeFirstName],[EmployeeMiddleName],[EmployeeLastName]

,[EmployeeGovernmentID],[DateOfBirth],[Department],[Manager]

from [dbo].[tblEmployeePart2]

--

select \*

from [dbo].[tblEmployeePart1]

INTERSECT

select \*

from [dbo].[tblEmployeePart2]

--

select \*

from [dbo].[tblEmployeePart1]

EXCEPT --MINUS

select \*

from [dbo].[tblEmployeePart2]

--

select \*

from [dbo].[tblEmployeePart2]

EXCEPT --MINUS

select \*

from [dbo].[tblEmployeePart1]

## Clustered and non-clustered indexes

PRIMARY KEY - Unique Clustered Index - Sorts the Table.

UNIQUE KEY - Unique Non-Clustered Index - Does not Sort the Table.

Unique/[Not Unique]

Heap

1 d

2 e

3 f

4 g

5 h

6 i

Continued on page 2

Page 2

7 a

8 b

9 c

10 j

11 k

12 l

---

a 7, 99, 149

b 8

c 9

CREATE UNIQUE CLUSTERED INDEX Ind\_tblEmployeePart1\_EmployeeGovernmentID

ON [dbo].[tblEmployeePart1]([EmployeeGovernmentID])

CREATE CLUSTERED INDEX Ind\_tblEmployeePart2\_EmployeeGovernmentID

ON [dbo].[tblEmployeePart2]([EmployeeGovernmentID])

CREATE NONCLUSTERED INDEX Ind\_tblEmployeePart2\_EmployeeNumberAndFirstName

ON [dbo].[tblEmployeePart2]([EmployeeNumber],[EmployeeFirstName])

scan -- slow

seek -- fast

SELECT [EmployeeFirstName]

FROM [dbo].[tblEmployee]

WHERE [EmployeeNumber] = 5

## Understand the need to secure a database

* Authentication
* Sensitive data
* Government regulation
* Prevent injection attacks
* Principal of least privilege

## User accounts and roles

* Logins allow access to the Instance, the SQL Server itself.
* Logins can use Windows Authentication (preferred) or SQL Server Authentication (for external logs, e.g. apps).
* Logins can use Server Roles, a set of preconfigured privileges. The widest Server Role is sysadmin.

## What objects can/should be secured

* The physical computer,
* The operating system (make sure you have security patches),
* The SQL Server program (with security patches),
* Tables, views, stored procedures, functions, triggers etc.

## What are backups?

Backups are a copy of your tables, queries, other objects and data, so you can restore it when you need it.

You can have multiple backups, and you can use multiple sets of hardware, both local (near your computer) and remote (for example, on the Internet).

## Backing up a database

--Full Everything

--Differential - Everything since the last Full backup.

--Transaction Log - Everything since the last backup.

--January 1 --Full

--Differential February 1: January 1 - February 1

--Differential March 1: January 1 - March 1

--Differential April 1: January 1 - April 1

--May 1 - Full

--Differential June 1: May 1 - June 1

--Full backup 12:00

--Transaction Log 1:00 12:00-1:00

--Transaction Log 2:00 1:00-2:00

--Differential 3:00 12:00-3:00

--Transaction Log 4:00 3:00-4:00

--Transaction Log 5:00 4:00-5:00

--Differential 6:00 12:00-6:00

--Transaction Log 7:00 6:00-7:00

--Transaction Log 8:00 7:00-8:00

Right-hand click on the database – Tasks – Back Up… - and select the Backup type.

## Restoring a database

Right-hand click on the database – Tasks – Restore – Database - and select:

* The appropriate files (full, differential, and transaction log(s)),
* Where do you want the database to be restored (into a new database, or on top of the existing one)?,
* Whether you want the tail-log to be backup, and
* Whether you want the source database to be in the restoring state (NORECOVERY). If so, you may need the following command afterwards:

RESTORE DATABASE [DatabaseFundamentals] WITH RECOVERY

GO