Add / Drop Constraints 10

add a constraint 10

ADD CONSTRAINT 10

Add multiple constraints to multiple columns 10

ALTER TABLE add the primary 12

constraints 10

CREATE TABLE 11

CREATE VIEWS 8

DELETE 7

delete / drop the constraint 10

DENY statement 9

Denying permissions 9

DROP CONSTRAINT 10

foreign key 12

Grant access to the database 8

GRANT statement 8

INSERT 12

INSERT INTO 12

INSERT statement 6

NESTED QUERIES 4

primary key 12

SELECT INTO 6

UPDATE MULTIPLE ROWS 7

UPDATE statement 7

UPDATE VIEWS 8

**Task 1.List the full names of all the library users**

select forename + ' ' + surname from Member

**Task 2. List the names of all the authors of books held by the library making sure that no duplicate names appear.**

Here you want to eliminate duplicate rows. So what you need to do is use the distinct function:

select distinct author from book

**Task 3.List the titles of books written by Graham Greene**

**(notice write as Greene Graham in SQL)**

Here you only use the WHERE function:

select title from book where author = 'Greene Graham'

**Task 4.List the book numbers and titles of all books acquired from 1991 onwards (the first two digits of bookno represent the acquisition year) (your answer need not be Y2K compliant!)**

Here use the WHERE function and comparison operator > :

select bookNo, title from book where bookNo>= 91000000

**Task 5. Who has written books about crochet ? (books with 'crochet' somewhere in the title)**

Here use the LIKE function %: select title from book where title like '%crochet%'

**Task 6. Repeat the list produced in question 3, but ensure that the output is in alphabetical order.**

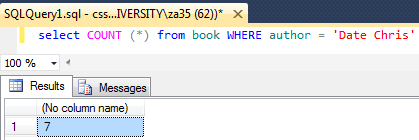
Here use the ORDER BY function:

select title from book where author = 'Greene Graham' order by title

**Task 7. How many books written by Chris Date are held in the library?**

Answer is 7. Here you use the COUNT(\*)function, together with the WHERE function:

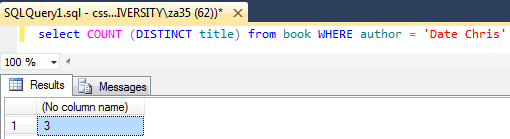
select COUNT(\*) from book WHERE author = 'Date Chris'



**Task 8. How many different titles written by Chris Date are held in the library?**

Answer is 3. Here you use the COUNT() function but pass in the DISTINCT function as a parameter, then use the WHERE function:

Select COUNT (DISTINCT title) from book WHERE author = 'Date Chris'



**Task 9. What is the class name of the only book in the library written by David Lodge?**

Here you need to join the table Classification with table Book, then select className WHERE author is Lodge David………………………

select className from classification

JOIN book on classification.classCode = book.classCode

WHERE author ='Lodge David'

**Task 10.Gadel has books on loan. When are they due back?**

select dueBack from loan

JOIN member on loan.memberNo = member.memberNo

WHERE surname = 'Gadel'

**Task 11. List the titles of all the books on loan to Bierpinski.**

Select title from book

JOIN loan on book.bookNo = loan.bookNo

JOIN member on loan.memberNo = member.memberNo

WHERE surname = 'Bierpinski'

**Task12. List the names of users with the titles of the books that they have on loan.**

Select forename + ' ' + surname, title from Member

JOIN loan on member.memberNo = loan.memberNo

JOIN book on loan.bookNo = book.bookNo

**Task 13. Repeat query 12, but this time you should also list those users who have no books on loan. (you'll need a special type of join for this)**

Select forename + ' ' + surname, title from Member

LEFT OUTER JOIN loan on member.memberNo = loan.memberNo

LEFT OUTER JOIN book on loan.bookNo = book.bookNo

-----------------------------------------------------------------------------------------

Week 7

OBS, making alias, example: select SUM(t.fines) from Member as t

Above is identical to select SUM(fines) from Member

The only reason you want to make an alias is when you are dealing with a very long query, so instead of keep on typing (t.fines), you can now refer to it as: t

**1. What is the total sum of fines owed to the library ?**

Select SUM(fines) from Member

**2. What is the average fine owed by members from the Philosophy department?**

Select AVG(fines) from member

WHERE dept = 'Philosophy'

**3. What is the earliest expiry date of any loan?**

Select MIN(dueBack) from loan

**4. Who has the books out which are about databases (the word 'database' is in the classname)?**

Select forename + ' ' + surname from member

JOIN loan on member.memberNo = loan.memberNo

JOIN book on loan.bookNo = book.bookNo

JOIN classification on book.classCode = classification.classCode

WHERE className like '%database%'

**5. List the users who owe fines - worst offenders first.**

Select forename + ' ' + surname, fines from member

WHERE fines > 0.00

ORDERBY fines DESC

You don’t really have to list the fines here, but I am only doing it to be able to see the result in the table, ie, fines are in descending order.

**NESTED QUERIES**

The result of one query can feed into another. Here is an example:

SELECT PACKID, PACKNAME

FROM PACKAGE

WHERE PACKCOST > (SELECT AVG (PACKCOST) FROM PACKNAME)

**6. Who owes more than 5% of the total of fines owed?**

Select forename + ' ' + surname

from member

WHERE fines > (select 0.05\*SUM(fines) from member)

**7. List any members of staff who have a fine which exceeds the average for Computing**select forename + ' ' + surname from member

WHERE fines > (select AVG(fines) from member WHERE dept = 'Computing' AND usertype = 'staff')

**8. What is the name of the user whose user ID is one of those with a loan to be returned on 8-Jun-2005? (Can you think of another way to answer this?)**

Without using sub-query:  
  
select DISTINCT (forename + ' ' + surname)

from member

JOIN loan on member.memberNo = loan.memberNo

WHERE dueBack = '8-Jun-2005'

------------------- but you should use sub-query:

select forename + ' ' + surname

from member

WHERE memberNo IN (select memberNo from loan WHERE dueBack = '8-Jun-2005')

**GROUP BY CLAUSE An example follows:**

SELECT TAGNUM, SUM (SOFTCOST)

FROM SOFTWARE

GROUP BY TAGNUM

**9. Show the total fines for each dept.**

select dept, SUM(fines) as Total\_Fines from member

GROUPBY dept

**10. Show the average fines for each dept.**

select dept, AVG(fines) as Average\_Fines from member

GROUPBY dept

**11. List how many books are on loan to each borrower.**

select memberNo, COUNT(bookNo) from loan

GROUPBY memberNo

If you apply criteria to a group rather than to each row you must use HAVING... rather than WHERE.....

**12 Show the classifications which are represented by more than 3 books.**

select className from Classification

JOIN Book

on Classification.classCode = Book.classCode

GROUPBY className

HAVING COUNT(bookNo) > 3

**Lab 3 EvenMoreSqlExercises**

**1. You can update tables for which you are either owner or have a permission to modify them. Using the SQL SELECT INTO statement, create your own book, member, loan and classification tables from the lend\_libdatabse.**

**(The SELECT INTO statement)**

Create the Loan table into my database za35:

SELECT \* INTO Loan

FROM lend\_lib.dbo.Loan;

Create the Book table into my database za35:

SELECT \* INTO Book

FROM lend\_lib.dbo.Book;

Create the Member into my database za35:

SELECT \* INTO Member

FROMlend\_lib.dbo.Member;

Create the Classificationtable into my database za35:

SELECT \* INTO Classification

FROM lend\_lib.dbo.Classification;

**2. Execute an INSERT statement to insert a new member**.

**(The INSERT statement)**

INSERT INTO za35.dbo.Member

(memberNo, surname, forename, telNo, dept, fines, usertype)

VALUES

(2212, 'Dany', 'Cox', 0123456789, 'Music', 2.22, 'staff')

Or

INSERT INTO Member

(memberNo, surname, forename, telNo, dept, fines, usertype)

VALUES

(3212, 'Dany', 'Cox', 0123456789, 'Music', 2.22, 'staff')

Or if you want to insert multiple rows in one go, then do as below:

INSERT INTO Member

(memberNo, surname, forename, telNo, dept, fines, usertype)

VALUES

(4212, 'Jane', 'Cox', 0123456789, 'Music', 2.22, 'staff'),

(5212, 'Dany', 'Cox', 0123456789, 'Music', 2.22, 'staff')

**3. Execute an UPDATE statement to change the details of a single member.**

**(The UPDATE statement)**

UPDATE Member

SET telNo = '07728626626'

WHERE

memberNo = 5212

or lets change the surename, telNo, dept and usertype:

UPDATE Member

SET surname = 'Daniels',

telNo = '07728626626',

dept = 'Computing',

usertype = 'student'

WHERE

memberNo = 5212

**(UPDATE MULTIPLE ROWS)**  
UPDATE Member

SET surname = 'Daniels',

telNo = '07728626626',

dept = 'Computing',

usertype = 'student'

WHERE

memberNo = 5212

UPDATE Member

SET surname = 'Daniels',

telNo = '07728626626',

dept = 'Computing',

usertype = 'student'

WHERE

memberNo = 4212

**4. Execute an UPDATE statement to apply an increase of 10% to all who have fines to pay.**

UPDATE Member

SET fines = fines \* 1.10

WHERE fines > 0.00

**5. Try to delete a book from the book table that has been on loan.**

DELETE FROM book

WHERE bookNo IN (select bookNo from loan

WHERE dueBack = '2005-06-25 00:00:00')

Or less than or equal:

DELETE FROM book

WHERE bookNo IN (select bookNo from loan

WHERE dueBack <= '2005-06-25 00:00:00')

**6. Create 5 views based on the previous sql statements you created. (CREATE VIEWS)**

CREATE VIEW vBook AS

SELECT \* FROM Book

**7. Try to update a table using one of your views.**

**(UPDATE VIEWS)**

You need to first create a view (as done above), then after you have created the view, then you can modify the view and run queries on it. Any modifications you make to the view will affect the underlying (main) tables.

*Example:*

UPDATE vBook

SET title = 'updating views',

author = 'Dany Cox'

WHERE

bookNo = 99657611

The above query will change the data in the main tableBook.

*Example 2*

First create view:  
CREATE VIEW vMember AS

SELECT \* FROM Member

Then modify the view  
INSERT INTO vMember

(memberNo, surname, forename, telNo, dept, fines, usertype)

VALUES

(1111, 'Dany', 'Cox', 0123456789, 'Music', 2.22, 'staff')

**8. Grant select access to one of your tables to a colleague (check that this has worked correctly)**

**(The GRANT statement)**

GRANT SELECT

ON Book TO "university\ws52"

You can also give them other rights such as select, update, insert, delete

GRANT select, update, insert, delete on dbo.Customer to "university\ws52"

You have now given them access to these tables, however, they cannot access these tables yet because they don’t have access to the database. You need to give them access to the database first so they then can access the individual tables. To give them access to the database, use below store procedure:

sp\_grantdbaccess 'university\ws52' **(Grant access to the database)**

**(The DENY statement) -** Denying permissions

Grant create table to Billy: GRANT CREATE TABLE TO "university\ws52"

Deny create table to Billy: DENY CREATE TABLE TO "university\ws52"

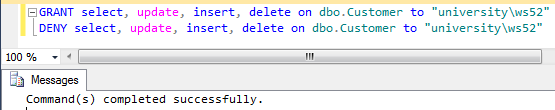
Grant billy access to Customer to perform select, update, insert, delete:

GRANT select, update, insert, delete on dbo.Customer to "university\ws52"

Deny billy access to Customer to perform select, update, insert, delete:

DENY select, update, insert, delete on dbo.Customer to "university\ws52"

**You can write both GRANT and DENY queries in one single query**

****

**9. Grant select access to one of your views, without granting access to the underlying tables (check that this one works correctly too).**

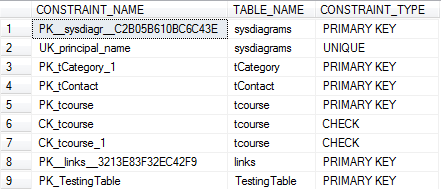
GRANT SELECT

ON vBook TO "university\ws52"

**10. Issue the following statements to see what constraints are in place for database tables.**

SELECT CONSTRAINT\_NAME, TABLE\_NAME, CONSTRAINT\_TYPE

FROM INFORMATION\_SCHEMA.TABLE\_CONSTRAINTS



**10. Using ALTER table add a CHECK CONSTRAINT to one of your tables and check that the constrain is working.**

**(Column check constraints) – Add / Drop Constraints**

Let’s create a table without constraints:

CREATE TABLE Persons

(

P\_Id int NOT NULL,

LastName varchar(255) NOT NULL,

FirstName varchar(255),

Address varchar(255),

City varchar(255),

PRIMARYKEY (P\_Id)

)

Let’s add a constraint to P\_Id

ALTER TABLE Persons

ADDCHECK (P\_Id>0)

BUT it is best to give the constraint a name so you can trace it if needed, like for example if you wanted to delete the constraint at later point, you just use the name of the constraint to drop it, as demonstrated below in DROP CONSTRAINT. For this constraint, I will give it the name chk\_Person:

ALTER TABLE Persons

ADD CONSTRAINT chk\_Person CHECK (P\_Id>0)

Let’s delete / drop the constraint named chk\_Person:

ALTER TABLE Persons

DROP CONSTRAINT chk\_Person

You could also write it simply as:

ALTER TABLE Persons

DROP chk\_Person

Add multiple constraints to multiple columns:

ALTER TABLE Persons

ADD CONSTRAINT chk\_Person CHECK (P\_Id > 0 AND City = 'Sandnes')

Above statement adds constraints to both the [P\_Id] and [City] columns.

Below query will add three constraints to the [P\_Id], [City] and [FirstName ] columns:

ALTER TABLE Persons

ADD CONSTRAINT chk\_Person CHECK (P\_Id > 0 AND City = 'Sandnes' AND FirstName = 'Dany')

**Create a table and set constraints at the same time:**

* **Option 1:** create table and set multiple constraints at the end of the query:

CREATE TABLE Persons

(

P\_Id int NOT NULL,

LastName varchar(255) NOT NULL,

FirstName varchar(255),

Address varchar(255),

City varchar(255),

CONSTRAINT chk\_Person CHECK (P\_Id > 0 AND City = 'Sandnes')

)

* **Option 2:** create table and add constrains next to each column you want to add constraints too:

CREATE TABLE Persons

(

P\_Id int NOT NULL CONSTRAINT chk\_Person CHECK (P\_Id > 0),

LastName varchar(255) NOT NULL,

FirstName varchar(255),

Address varchar(255),

City varchar(255) CONSTRAINT chk\_City CHECK (City = 'Sandnes'),

)

**11. Execute CREATE TABLE statement to create tables for the following relations:**

a)

**Departments** ( deptCode SMALLINT,

deptName VARCHAR(30),

onFloor TINYINT)

CREATE TABLE Departments

(deptCode SMALLINT NOT NULL,

deptName VARCHAR(30),

onFloor TINYINT

)

b)

**Employee** ( empID SMALLINT,

empSurname VARCHAR(30),

empForename VARCHAR(20),

dateStarted SMALLDATETIME,

salary SMALLMONEY,

dept SMALLINT )

CREATE TABLE Employee

(empID SMALLINT NOT NULL,

empSurname VARCHAR(30),

empForename VARCHAR(20),

dateStarted SMALLDATETIME,

salary SMALLMONEY,

dept SMALLINT

)

**Using ALTER TABLE add the primary key and the foreign key constraints to these tables.**

**(ALTER TABLE add the primary)**

1. Add the primary key pk\_deptCode to table Departments

ALTERTABLE Departments

ADD CONSTRAINT pk\_deptCode PRIMARYKEY (deptCode)

1. Add the primary key pk\_empID to table Employee

ALTERTABLE Employee

ADD CONSTRAINT pk\_empID PRIMARYKEY (empID)

Add the foreign key fk\_deptCode to table Employee

ALTERTABLE Employee

ADD CONSTRAINT fk\_deptCode

FOREIGNKEY (dept)

REFERENCES Departments(deptCode)

<http://www.w3schools.com/sql/sql_foreignkey.asp> (add create table and foreign key….)

**12. Insert several records using SQL statements into each of the table.**

Let’s insert some data into Departments

INSERT INTO Departments

(deptCode, deptName, onFloor)

VALUES

(1, 'Computing', 3),

(2, 'Maths', 5)

Let’s insert some data into Employee

INSERT INTO Employee

(empID, empSurname, empForename, dateStarted, salary, dept)

VALUES

(1, 'Smith', 'Mike', '2013/05/12', 67000, 1),

(2, 'Michaels', 'Dany', '2013/04/12', 17000, 2)

**Lab 5 ci204PhysicalStorageExercises\_2012-2013**

**1**. **Make sure you are using your own database first and delete the old table if exists:**

USE {your own database}

GO

DROP TABLE links;

GO

**2. Create a table with primary key:**

CREATE TABLE links(

Id INT PRIMARYKEY,

url VARCHAR(80) NOT NULL,

notes VARCHAR(1024),

counts INT,

created DATETIME NOT NULL DEFAULT(getdate())

); GO

**3. If you want to know how many indexes have been defined for a given table, you can use the SP\_HELP built-in stored procedure (SP) in the following syntax:**

EXEC SP\_HELP table\_name

-- Returns all database objects related the given table

Execute the following SP statement:

EXEC SP\_HELP links;

GO

OBS: There could be only one clustered index per table. If you have a primary key, then it will become the clustered index, otherwise you can chose other data as clustered index, but only one can be!

OBS: clustered index = unique index

OBS: you can have multiple non-clustered indexes per table

**4. Create a new table without the primary key for which you will create two indexes:**

CREATETABLE links\_indexed(

id INT,

url VARCHAR(80)NOT NULL,

notes VARCHAR(1024),

counts INT,

created DATETIME NOT NULL DEFAULT(getdate())

);

GO

**5. Now, create three indexes on the fields** url**,** counts **and** id**:**

CREATE INDEX idx\_links\_url ON links\_indexed(url);

CREATE INDEX idx\_links\_counts ON links\_indexed(counts);

CREATE INDEX idx\_links\_id ON links\_indexed(id);

GO

**6. Using the store procedure from the exercise 3, see how many indexes have been created on table links\_indexed.**

EXEC SP\_HELP links\_indexed;

GO

**7. Another way to view existing indexes defined for a given table is to use the system view called** "sys.indexes"**. The following statement shows you how many indexes were defined from the previous exercise on table** "links\_indexed"**:**

SELECT \* FROM sys.indexes WHERE object\_id=(

SELECT object\_id FROM sys.tables WHERE name = 'links\_indexed'

);

GO

**8. To remove an index, use DROP INDEX table\_name.index\_name. The following statement removes index** idx\_links\_id **from the table** *links\_indexed***:**

DROP INDEX links\_indexed.idx\_links\_id;

**9. An index for multiple columns works similarly to a sorting process on multiple columns. If an index is defined for two columns, the index key is composed by values from those two columns. A multi-column index will be used to speed up the search process based on the same columns in the same order as the index definition. For example, if you define an index called "combo\_index" for "url" and "counts". "combo\_index" will be used only when searching or sorting rows by "url" and "counts".**

**The following exercise shows you how to create an index for two columns:**

CREATE INDEX combo\_index ON links\_indexed(url,counts);

GO

**View indexes:**

EXEC SP\_HELP links\_indexed;

**Task B FURTHER EXERCISES**

Performance test of INSERT statement

When indexes are defined for a table, each time a new row is inserted to the table, all the indexes must be updated. This extra update work could slow down the insert statement. To find out how much slower the INSERT statements will be on tables with indexes, you need first to measure the performance of INSERT statements on tables without indexes and then on tables with indexes.

**1. First, create table Cust from the table gs3.Customer:**

Select \* into Cust from gs3.dbo.Customer

**Empty the table**

delete from Cust;

**2. Measure time for inserting records.**

DECLARE @start\_time DATETIME, @end\_time DATETIME;

SET @start\_time = GETDATE();

INSER TINTO Cust

SELECT CustomerID, FirstName, LastName, City

FROM gs3.dbo.Customer

SET @end\_time = GETDATE();

PRINT 'Milliseconds used: ' + CONVERT(VARCHAR(20),

DATEDIFF(MILLISECOND, @start\_time, @end\_time));

GO

Click on the Messages tab to see how long it took to insert the records.

Messages tab shows: (18492 row(s) affected)

Milliseconds used: 63

**3. Now create a table Cust\_indexed, which will be indexed:**

Select \* into Cust\_indexed from gs3.dbo.Customer

**Empty the table:**

delete from Cust\_indexed;

**4. Now create two indexes on Cust\_indexed:**

CREATE INDEX idx\_cust\_FirstName ON Cust\_indexed(FirstName);

CREATE INDEX idx\_cust\_LastName ON Cust\_indexed(LastName);

GO

**Measure the time for inserting records into indexed table:**

DECLARE @start\_time DATETIME, @end\_time DATETIME;

SET @start\_time = GETDATE();

INSERT INTO Cust\_indexed

SELECT CustomerID, FirstName, LastName, City

FROM gs3.dbo.Customer

SET @end\_time = GETDATE();

PRINT 'Milliseconds used: ' + CONVERT(VARCHAR(20),

DATEDIFF(MILLISECOND, @start\_time, @end\_time));

GO

Click on the Messages tab to see how long it took to insert the records.

Messages tab shows: (18492 row(s) affected)

Milliseconds used: 1080

**Does Index Speed Up SELECT Statements?**

If you want to see the impact of indexes on SELECT statements, you can run the same

SELECT statement on "Cust” and “Cust\_indexed" tables:

**5. Run SELECT on the table without indexes:**

DECLARE @start\_time DATETIME, @end\_time DATETIME;

SET @start\_time = GETDATE();

SELECT TOP 3 FirstName, LastName FROM Cust

WHERE City LIKE 'L%'

ORDERBY LastName DESC;

SET @end\_time = GETDATE();

PRINT 'Milliseconds used: ' + CONVERT(VARCHAR(20),

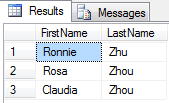
DATEDIFF(MILLISECOND, @start\_time, @end\_time));

GO

Click on the Messages tab to see how long it took to insert the records.

Messages tab shows: (3 row(s) affected)

Milliseconds used: 6



**6. Repeat the same SELECT statement on the table with indexes and compare the results.**

DECLARE @start\_time DATETIME, @end\_time DATETIME;

SET @start\_time = GETDATE();

SELECTTOP 3 FirstName, LastName FROM Cust\_indexed

WHERE City LIKE 'L%'

ORDERBY LastName DESC;

SET @end\_time = GETDATE();

PRINT 'Milliseconds used: ' + CONVERT(VARCHAR(20),

DATEDIFF(MILLISECOND, @start\_time, @end\_time));

GO

Click on the Messages tab to see how long it took to insert the records.

Messages tab shows: (3 row(s) affected)

Milliseconds used: 0

**SQL Alias**

You can give a table or a column another name by using an alias. This can be a good thing to do if you have very long or complex table names or column names.

An alias name could be anything, but usually it is short.

SQL Alias Syntax for Tables

SELECT column\_name(s)  
FROM table\_name  
AS alias\_name

SQL Alias Syntax for Columns

SELECT column\_name AS alias\_name  
FROM table\_name

**Alias Example**

Using the "Customers" and "Orders" tables, we give the table aliases of "c" and "o" respectively.

The following SQL statement selects all the orders from the customer "Alfreds Futterkiste":

SELECT o.OrderID, c.CustomerName  
FROM Customers AS c,  
Orders AS o  
WHERE c.CustomerName='Alfreds Futterkiste';

Or above query can be simplified by removing the AS keyword:

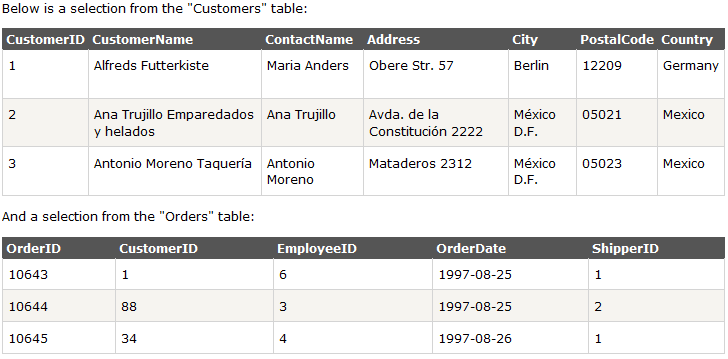
SELECT o.OrderID, c.CustomerName  
FROM Customers c,  
Orders o  
WHERE c.CustomerName='Alfreds Futterkiste';

Below is the same SELECT statement as above without aliases:

SELECT Orders.OrderID, Customers.CustomerName  
FROM Customers,  
Orders  
WHERE Customers.CustomerName='Alfreds Futterkiste';

<http://www.w3schools.com/sql/sql_foreignkey.asp>

**Writing joins in two different ways:**



Let’s do the below join: **Example 1**

SELECT ShipperID

FROM Customers

INNER JOIN Orders on Customers.CustomerID = Orders.CustomerID

The above join can be rewritten as below: (note the below query is using alias)

SELECT o.ShipperID

FROM Customers c, Orders o

WHERE c.CustomerID = o.CustomerID

**Example 2**

SELECT CustomerName, ShipperID

FROM Customers

INNER JOIN Orders on Customers.CustomerID = Orders.CustomerID

WHERE Country = 'Mexico'

The above join can be rewritten as below: (note the below query is using alias)

SELECT c.CustomerName, o.ShipperID

FROM Customers c, Orders o

WHERE c.CustomerID = o.CustomerID

AND Country = 'Mexico'

The previous query from example 2 written without alias:

SELECT c.CustomerName, o.ShipperID

FROM Customers c, Orders o

WHERE Country = 'Mexico'

AND c.CustomerID = o.CustomerID

SELECT CustomerName, ShipperID

FROM Customers, Orders

WHERE Customers.CustomerID = Orders.CustomerID

AND Country = 'Mexico'

The previous query from example 2 can also be written as below:

Summary of key queries:

How to create a table, primary key, foreign key, and add multiple constraints all simultaneously in one query:

CREATE TABLE Employees (

empID SMALLINT NOT NULL PRIMARY KEY,

empSurname VARCHAR(30),

empForename VARCHAR(20),

dateStarted SMALLDATETIME

Adding constraints

CONSTRAINT startDateCheck

CHECK (dateStarted >= 'May 17, 1998'),

salary SMALLMONEY

Adding constraints

CONSTRAINT minimumSalary

CHECK (salary >= €20000),

dept SMALLINT

Adding FK form the Departments table

REFERENCES Departments(deptCode)

)

------------------------------------------------------------------------------------------------------------------------------

How to create a compound key using sql ddl:

firstPart SMALLINT NOT NULL,

secondPart CHAR(6) NOT NULL,

Adding compound key

**PRIMARY KEY (firstPart, secondPart)**

How to create a compound foreign key using sql ddl:

…

firstPart SMALLINT,

secondPart CHAR(6),

Adding compound foreign key

**FOREIGN KEY (firstPart, secondPart)** REFERENCES

MasterTable(firstPart, secondPart)

)