Naam: Matthias van Schoonhoven

Studentnummer: 170962

Datum: 1-6-2020

Versie 3

Databases

Inhoud

[Inleiding 2](#_Toc41942032)

[Deelopdracht 1: 2](#_Toc41942033)

[Deelopdracht 2 3](#_Toc41942034)

[Deelopdracht 3 6](#_Toc41942035)

[Deelopdracht 4 9](#_Toc41942036)

[Deelopdracht 5 10](#_Toc41942037)

[Deelopdracht 6 10](#_Toc41942038)

[Deelopdracht 7 12](#_Toc41942039)

# Inleiding

Dit blok zijn we bezig geweest met sql queries en het ontwerpen van een database. In week 1 tot en met 3 draaide het vooral om het maken van queries. Week 4 ging het over powerbi en het maken van ontwerpen voor een database. Week 5 over het deleten, updaten en invoeren van data. Week 6 over het zelf maken van een nieuwe database en week 7 over het beter maken van een database. Door onhandigheden eruit te halen en de database te verbeteren.

# Deelopdracht 1:

In week 1 zijn we begonnen met onze eerste queries te schrijven. Deze queries bestonden alleen nog maar uit SELECT, FROM en WHERE. SELECT word gebruikt bij het selecteren van data uit de database.

FROM is uit welke table je de data haalt en WHERE gebruik je om een conditie toe te voegen. Zoals genre = shooter. Wat alleen resultaten geeft die als genre shooter hebben.

/\*\* query 1 - Matthias van Schoonhoven

Beschrijving: Een overzicht van shooter spellen

\*\*/

SELECT \*

FROM games

WHERE genre = 'shooter'

/\*\* query 2 - Matthias van Schoonhoven

Beschrijving: overzicht van alle spellen met als platform de ps4

\*/

SELECT \*

FROM games

WHERE platform = 'ps4'

/\*\* query 3 - Matthias van Schoonhoven

Beschrijving: overzicht van alle spellen met als publisher nintendo

\*/

SELECT \*

FROM games

WHERE publisher = 'nintendo'

/\*\* opdracht 4 Alle spellen met het jaar 2006

\*\*/

SELECT \*

FROM games

WHERE year = '2006'

/\*\* opdracht 5 Alle spellen uit het huidige jaar (dus geen '2019' gebruiken!)

\*\*/

SELECT \*

FROM games

WHERE year =

/\*\* opdracht 6 Alle spellen met in de naam het woord 'Tetris' (moet 29 resultaten opleveren!!).

\*\*/

Select \*

FROM games

WHERE name like '%tetris%'

/\*\* opdracht 7 De spellen die in de TOP-10 staan (rank 1 tot en met 10).

\*\*/

SELECT TOP 10 \*

FROM games

/\*\* opdracht 8Spel uit de TOP-10 OF het 100e spel uit de lijst (Battlefield 3), in totaal dus 11 spellen te tonen (de top 10 én het 100e spel)

\*\*/

SELECT TOP 10 \*

FROM games

SELECT \*

FROM games

WHERE rank = '100'

/\*\* opdracht 9 Toon alle spellen die meer dan 80 hebben omgezet

\*\*/

SELECT \*

FROM games

WHERE global\_sales > '80'

/\*\* opdracht 10 extended - geef per regio het 'percentage' weer van de omzet per regio!

\*\*/

SELECT Global\_sales, ROUND(CAST(NA\_sales \* 100.0 / global\_sales as FLOAT), 1) as NA\_sales,

ROUND(CAST(EU\_sales \* 100.0 / global\_sales as FLOAT), 1) as EU\_sales,

ROUND(CAST(JP\_Sales \* 100.0 / global\_sales as FLOAT), 1) as JP\_sales,

ROUND(CAST(Other\_Sales \* 100.0 / global\_sales as FLOAT), 1) as Other\_sales

FROM games

# Deelopdracht 2

In week 2 werden de queries al iets moeilijker en kwamen GROUP BY, HAVING en ORDER BY er ook bij. GROUP BY groepeert rijen die dezelfde value hebben bij elkaar. Bijvoorbeeld tel het aantal klanten uit elk land. Bij GROUP BY word vaak ook gebruikt gemaakt van COUNT, MAX, MIN, SUM en AVG. HAVING word gebruikt, omdat je WHERE niet gebruikt kan worden bij aggregate functions.

ORDER BY word gebruikt bij het in volgorde zetten van de resultaten.

/\*\* query 4 - Matthias van Schoonhoven \*\*/

SELECT Platform, COUNT(id) AS Aantal\_spellen\_platform

FROM Games

WHERE genre = 'sports'

GROUP BY Platform

HAVING Platform IN ('wii', 'PS3', 'PS4')

ORDER BY Aantal\_spellen\_platform ASC

/\*\* query 5 - Matthias van Schoonhoven \*\*/

SELECT year, COUNT(id) AS Aantal\_spellen\_jaar

FROM Games

WHERE genre = 'racing'

GROUP BY year

HAVING year IN ('2005','2006','2007','2008')

ORDER BY Aantal\_spellen\_jaar ASC

/\*\* query 6 - Matthias van Schoonhoven \*\*/

SELECT publisher, COUNT(id) AS aantal\_spellen-publisher

FROM Games

WHERE year = '2006'

GROUP BY publisher

HAVING publisher IN ('Nintendo','ELectronic Arts')

ORDER BY aantal\_spellen\_publisher ASC

/\*\* query 7 - De query toont het overzicht van alle spellen die het woord 'Sports' in de naam hebben\*\*/

SELECT Name

FROM Games

WHERE name LIKE '%sports%'

/\*\* query 8 -De query toont de Rank, het platform, het ID en de naam van alle Nintendo spellen uit de jaren 90 (1990 t/m 1999) en de 'Global\_Sales' > 10 (totaal aantal resultaten: 7!!) \*\*/

SELECT Rank, platform, id, name

FROM games

WHERE publisher = 'Nintendo'

AND year BETWEEN '1990' AND '1999'

AND Global\_sales > 10

/\*\* query 9 -Toon de totale omzet van alle spellen per Publisher per regio (579 regels!!) \*\*/

SELECT publisher, COUNT(global\_sales) as TO\_global\_sales

FROM games

GROUP by publisher

ORDER BY TO\_global\_sales ASC

/\*\* query 10 \*\*/

/\*\*

a. Wat is hoogste behaalde omzet in Japan (10.22), en welke query gebruik je daarvoor?\*\*/

SELECT MAX(JP\_sales) AS hoogste\_omzet\_Japan

FROM GAMES

/\*\*

b. Neem het resultaat van 'a' mee in de WHERE clause van de volgende vraag: Bij welk spel hoort de JAPANSE Omzet van '10.22' \*\*/

SELECT name

FROM GAMES

WHERE JP\_sales = '10.22'

/\*\*

c. EXTENDED: kun je dit ook in één query oplossen?\*\*/

SELECT name, jp\_sales

FROM GAMES

where jp\_sales = (

SELECT MAX(jp\_sales)

FROM GAMES

)

/\*\* query 11 Wat is het goedkoopste product in de database w3schools?

En welke query gebruik je hiervoor? \*\*/

SELECT ProductName, price

FROM products

WHERE price = (

SELECT MIN(price)

from products

)

/\*\* query 12 De overstap naar het 'duurste' product moet op basis van query 11 niet zo moelijk zijn. Moeilijker is het om te bepalen welke details je kunt opvragen over dit product? \*\*/

SELECT ProductName, price

FROM products

WHERE price = (

SELECT MAX(price)

from products

)

/\*\* 13 - Geef de query om te laten zien dat order 10248 bestaat uit 12 x product 11 en 10 x product 42 en 5 x product 72). \*\*/

SELECT \*

FROM order\_details

WHERE OrderID = '10248'

/\*\* query 14 EXTENDED Bepaal nu met behulp van één SQL Statement de 'totaalwaarde' van order: 10248 \*\*/

SELECT SUM(products.Price \* order\_details.Quantity)

from Order\_details INNER JOIN Products

ON Order\_details.ProductID = Products.ProductID

WHERE OrderID = '10248'

/\*\* en de query wordt alleen maar eenvoudiger wanneer de vraag is om de totale omzet uit te rekenen: [386424.23], welke query doet dit dan? \*\*/

SELECT SUM(products.Price \* order\_details.Quantity)

from Order\_details INNER JOIN Products

ON Order\_details.ProductID = Products.ProductID

/\*\* 15 - Wat is de gemiddelde omvang (quantiy) van orders binnen W3SCHOOLS? \*\*/

SELECT AVG(quantity)

FROM Order\_details

/\*\* 16 - Hoeveel verschillende product zijn er in de database én welke query gebruik je daarvoor? \*\*/

SELECT COUNT(ProductID)

FROM products

/\*\* 17 - Hoeveel verschillende producten zijn er gebruikt in de orderdatabase, met behulp van welke query toon je dit aan?\*\*/

SELECT COUNT(DISTINCT productid)

FROM Order\_details

/\*\* 18 EXTENDED - Welke producten zijn nog nooit besteld?

En met welke query toon je dat aan?\*\*/

SELECT \*

FROM Order\_details

WHERE Order\_details.ProductID NOT IN (

SELECT ProductID

FROM products

)

/\*\* 19 Wat is de totale waarde van producten in W3schools, ervanuit gaande dat van ieder product maar één product aanwezig is?) \*\*/

SELECT SUM(price)

FROM products

/\*\* 20, leg de foutmelding uit: Met de SELECT \* krijg je alles, maar Games.ID die je dan dus ook krijgt bevat de GROUP BY clause niet. \*\*/

/\*\* 21 - Welke query heb ik nodig om de volgende tabel te realiseren (op de games database) \*\*/

SELECT Genre, COUNT(RANK) as Aantal\_spellen, SUM(Global\_sales) as Omzet

FROM Games

GROUP BY GENRE

/\*\* 22 - laat nu alleen de genres zien waar er meer dan 900 van zijn en minder dan 1000 'omzet' (dus alleen de 'misc', 'adventure', 'racing' en 'role-playing' resultaten). \*\*/

SELECT Genre, COUNT(RANK) as Aantal\_spellen, SUM(Global\_sales) as Omzet

FROM Games

GROUP BY GENRE

HAVING COUNT(RANK) > 900

AND SUM(global\_sales) < 1000

# Deelopdracht 3

In week 3 werden de queries echt al een stuk uitdagender en kwamen JOINS veel voor.

Een JOIN gebruik wanneer je meerdere rijen van 2 of meer tabellen wilt samenvoegen, gebaseerd op een gerelateerde kolom hier tussen.

/\*\* 1. Geef een overzicht van alle berijders \*\*/

SELECT \*

FROM berijder

/\*\* 2. Geef een overzicht van alle contracten \*\*/

SELECT \*

FROM contract

/\*\* 3. Geef een overzicht van alle berijders, gecombineerd met de contracten (verklaar het aantal 'rows') \*\*/

SELECT \*

FROM Contract, Berijder /\*\* een berijder heeft meerdere contracten \*\*/

/\*\* 4. Geef een overzicht van alle berijders met een contract bij 'Arval bv' \*\*/

SELECT \*

FROM Contract, Berijder

WHERE LeaseMaatschappijId = '4'

/\*\* 5. Geef een overzicht van alle auto's die meneer 'Walt van Maas' ooit geleased heeft \*\*/

SELECT \*

FROM contract

INNER JOIN Auto

ON contract.autoid = auto.id

WHERE Berijderid = '9'

/\*\* 6. Maak een query die laat zien hoeveel leaseauto's ooit verleased zijn door 'Arval bv' \*\*/

Select COUNT(leasemaatschappijid)

FROM contract

WHERE Leasemaatschappijid = '4'

/\*\* 7. Geef een overzicht van alle berijders met een nu lopend contract (GETDATE() geeft de huidige datum) \*\*/

SELECT \*

FROM contract

WHERE EindDatum >= GETDATE()

/\*\*. 8. Geef een overzicht van alle auto's die wel in het systeem zitten, maar nooit verleased zijn. (Bij welke leasemaatschappij zijn die auto's opgevoerd?). \*\*/

SELECT \*

FROM contract RIGHT OUTER JOIN auto

ON contract.autoid = auto.id

WHERE contract.id IS NULL

/\*\* 9. Geef een overzicht met alle contracten met een berijder waar het rijbewijs onbekend is. \*\*/

SELECT \*

FROM contract INNER JOIN berijder

ON contract.berijderid = berijder.id

WHERE RijbewijsNummer is NULL

/\*\* 10. Beschrijf in NATUURLIJKE taal wat het resultaat van onderstaande query is:

SELECT l.Naam, Merk, Count(c.Id) as aantalAutos

FROM contract c

INNER JOIN auto a ON a.Id = c.AutoId

INNER JOIN LeaseMaatschappij l ON l.id = c.LeaseMaatschappijId

GROUP BY l.Naam, a.Merk

HAVING SUM(a.Catalogusprijs) > 30000

ORDER BY count(a.Id) DESC

Laat de naam, merk en het aantal auto's zien met een gezamelijke prijs hoger dan 30000

En wat is het verschil met:

SELECT l.Naam, Merk, Count(c.Id) as aantalAutos

FROM contract c

INNER JOIN auto a ON a.Id = c.AutoId

INNER JOIN LeaseMaatschappij l ON l.id = c.LeaseMaatschappijId

WHERE a.Catalogusprijs > 30000

GROUP BY l.Naam, a.Merk

ORDER BY count(a.Id) DESC

Het verschil is dat bij de 2de query naar de auto's met een catalogusprijs van meer dan 30000 word gekeken en bij de 1ste query naar de gezamelijke prijs van meer dan 30000 gekeken

\*\*/

/\*\* 11. Wat is de naam van de oudste lease-auto bestuurder? \*\*/

SELECT voornaam, tussenvoegsel, achternaam, geboortedatum

FROM Berijder

WHERE geboortedatum = (

SELECT MAX(geboortedatum)

FROM Berijder

)

/\*\* 12. Wat is de naam van de jongste lease-auto bestuurder? \*\*/

SELECT voornaam, tussenvoegsel, achternaam, geboortedatum

FROM Berijder

WHERE geboortedatum = (

SELECT MIN(geboortedatum)

FROM Berijder

)

/\*\* 13. Welke berijders hebben helemaal geen contract? \*\*/

SELECT voornaam

FROM berijder

WHERE NOT EXISTS (

SELECT berijderid

FROM Contract, berijder

WHERE berijder.id = contract.berijderid

)

/\*\* 14. Hoeveel contracten zijn er afgesloten per leasemaatschappij? \*\*/

SELECT COUNT(contract.LeaseMaatschappijId) as Aantal\_contracten, LeaseMaatschappij.naam

FROM Contract

INNER JOIN LeaseMaatschappij

ON Contract.LeaseMaatschappijId = Leasemaatschappij.id

GROUP BY LeaseMaatschappij.naam

/\*\* 15. Wat is de gemiddelde prijs van leaseauto's per berijder? \*\*/

SELECT Contract.berijderid, AVG(catalogusprijs) AS gemiddelde\_prijs\_leaseauto

FROM contract

INNER JOIN auto

ON contract.autoid = auto.id

GROUP BY contract.berijderid

/\*\* 16. Wat is de gemiddelde prijs van leaseauto's per leasemaatschappij? \*\*/

SELECT AVG(catalogusprijs) AS gemiddelde\_prijs\_leaseauto, Leasemaatschappij.naam

FROM contract

INNER JOIN auto

ON contract.autoid = auto.id

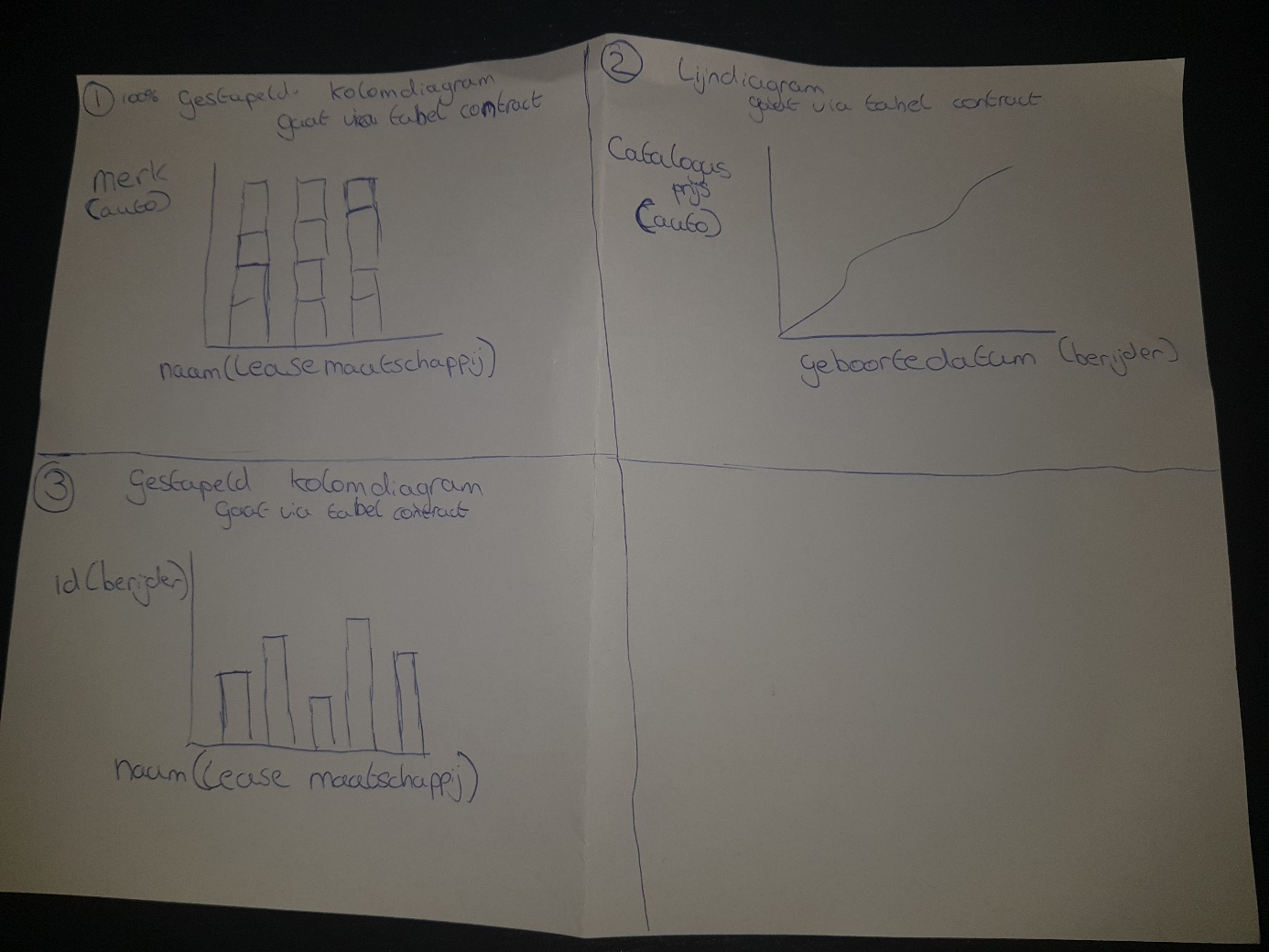
INNER JOIN leasemaatschappij

ON contract.leasemaatschappijid = leasemaatschappij.id

GROUP BY leasemaatschappij.naam

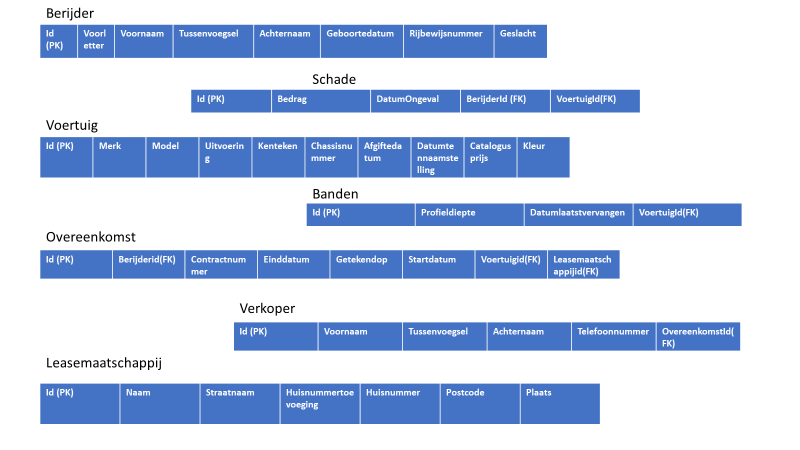
# Deelopdracht 4

In week 4 kwam het gebruik van PowerBi erbij. In powerbi kan je van gegevens overzetten naar inzichten en dat allemaal in 1 overzicht. In deze week moeten we ook zelf een paar ontwerpen maken die betrekking hebben op de lease database.



# Deelopdracht 5

In week 5 ging het vooral om het uitbreiden of het wijzigen van een database. We maakte veel gebruik van UPDATE, DELETE en INSERT.



Ik heb ervoor gekozen om een tabel schade toe te voegen. Dit zodat de opdrachtgever kan zien of berijder al schade heeft opgelopen. In deze tabel kan je zien welke voertuig schade heeft opgelopen, het bedrag en het datum van het ongeval. Ook vond ik het handig voor de opdrachtgever om te zien hoe de banden van de auto’s ervoor staan. In deze tabel staat beschreven wat de profieldiepte van de band is en wanneer deze voor het laatst is vervangen. Dit zodat de opdrachtgever kan zien of de band weer toe is aan vervanging en hoe vaak dit ongeveer moet gebeuren. Ook heb ik er voor gekozen om een tabel toe te voegen waar de verkoper van de auto(het leasecontract) is te vinden. In de tabel is de voornaam, tussenvoegsel en achternaam te vinden van de verkoper en het telefoonnummer zodat er contact kan worden opgenomen met de verkoper.

# Deelopdracht 6

In week 6 ging het om het zelf maken van een nieuwe database. Ook ging het over foreign keys en primary keys. Een foreign key word gebruikt bij het linken van twee tabellen met elkaar. Een foreign key is een veld of een verzameling van velden die in 1 tabel verwijst naar de primary key die in een andere tabel staat.

/\*\*opdracht 6b Schrijf een create-statement waarmee je de Student-tabel maakt met de volgende kolommen: \*\*/

CREATE TABLE [Student]

(

[Id] [int] NOT NULL PRIMARY KEY,

[StudentNummer] [int] NOT NULL,

[Voornaam] [nvarchar](100) NOT NULL,

[Tussenvoegsel] [nvarchar](15) NULL,

[Achternaam] [nvarchar](100) NOT NULL,

[Geboortedatum] [date] NULL,

)

/\*\*opdracht 6c Maak nu een alter-table script waarmee je een constraint toevoegt: dwing af dat studcentnummer uniek moet zijn. \*\*/

ALTER TABLE Student

ADD CONSTRAINT UC\_Student UNIQUE (StudentNummer);

/\*\*OPDRACHT 6d Schrijf een create-statement waarmee je de Vak-tabel maakt met de volgende kolommen: \*\*/

CREATE TABLE [Vak]

(

[Id] [int] NOT NULL PRIMARY KEY,

[VakCode] [nvarchar](10) NOT NULL,

[Naam] [nvarchar](255) NOT NULL,

[Jaar] [int] NULL,

)

/\*\*opdracht 6e Maak nu nog een alter-table script waarmee je een constraint toevoegt: dwing af dat de combinatie van vakcode en jaar uniek is. \*\*/

ALTER TABLE Vak

ADD CONSTRAINT uc\_vak UNIQUE (VakCode, Jaar);

/\*\*OPDRACHT 6f Schrijf een create statement waarmee de Cijfertabel aanmaakt met de volgende kolommen:\*\*/

CREATE TABLE [Cijfertabel]

(

[Id] [int] NOT NULL PRIMARY KEY,

[VakId] [int] NOT NULL,

[StudentId] [int] NOT NULL,

[Cijfer] [int] NOT NULL,

[DatumBehalen] [date] NOT NULL,

[IsHerkansing] [bit] NOT NULL,

)

/\*\*OPDRACHT 6g Voeg nu de 2 foreign-key relaties uit het bovenstaande schema toe.

Gedaan in table designer en dan relationships \*\*/

/\*\* opdracht 6h Schrijf een create-table statement dat de Docent-tabel toevoegt met onderstaande kolommen. \*\*/

CREATE TABLE [Docent]

(

[Id] [int] NOT NULL PRIMARY KEY,

[Email] [nvarchar](50) NOT NULL,

[Voornaam] [nvarchar](100) NOT NULL,

[Tussenvoegsel] [nvarchar](15) NULL,

[Achternaam] [nvarchar](100) NOT NULL,

)

/\*\* opdracht 6i Breid nu de vak-tabel uit met een ExaminatorId en zet hier een foreign-key relatie op naar de docent-tabel. \*\*/

ALTER TABLE vak

ADD ExaminatorId int;

/\*\* OPDRACHT 6j Schrijf een script dat jaar verwijdert uit de vak-tabel. \*\*/

ALTER TABLE vak

DROP CONSTRAINT uc\_vak;

ALTER TABLE vak

DROP COLUMN jaar;

/\*\*OPDRACHT 6k Schrijf een script dat zowel een startdatum (type date, nullable) en een schooljaar (nvarchar(9), not nullable) toevoegt aan de vak-tabel. Voeg ook een nieuwe logische unique-constraint toe. \*\*/

ALTER TABLE vak

ADD Startdatum [date] null;

ALTER TABLE VAK

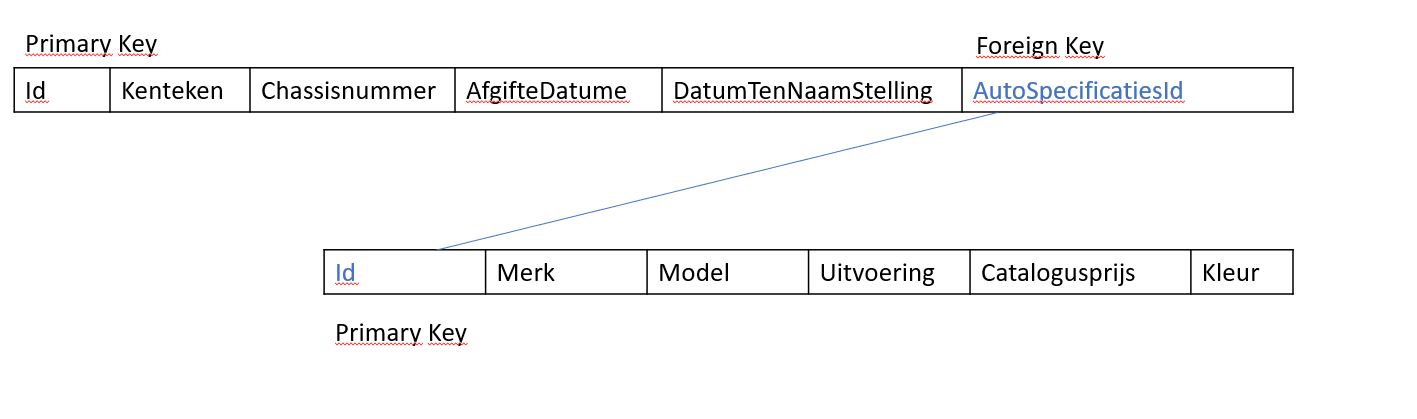
ADD Schooljaar [nvarchar](9) not null;

ALTER TABLE Vak

ADD CONSTRAINT uc\_vak UNIQUE (VakCode, Schooljaar);

# Deelopdracht 7

Bij de afronderende opdracht ging het er vooral om dat sommige tabellen op een onhandige manier zijn ontworpen en was de opdracht dus om de auto tabel uit elkaar te trekken in meerdere tabellen. Hierbij moest eerst een ontwerp gemaakt worden en daarna aan de slag in sql met het maken van nieuwe tabellen, toevoegen van key en constraints en natuurlijke het verwijderen van oude kolommen.



/\*\* 2\*\*/

CREATE TABLE [AutoSpecificaties]

(

[Id] [int] IDENTITY(1,1) NOT NULL,

[Merk] [nvarchar](150) NOT NULL,

[Model] [nvarchar](150) NOT NULL,

[Uitvoering] [nvarchar](150) NOT NULL,

[Catalogusprijs] [decimal](18, 2) NOT NULL,

[Kleur] [nvarchar](150) NOT NULL,

CONSTRAINT [PK\_AutoSpecificaties] PRIMARY KEY ([Id])

)

/\*\* 3 \*\*/

insert into AutoSpecificaties

(Merk, Model, Uitvoering, Catalogusprijs, Kleur)

(select distinct Merk, Model, Uitvoering, Catalogusprijs, Kleur From Voertuig)

/\*\* 4 \*\*/

ALTER TABLE Voertuig

ADD AutoSpecificatiesId int;

ALTER TABLE Voertuig

ADD CONSTRAINT FK\_AutoAutoSpecs

FOREIGN KEY (AutoSpecificatiesId) REFERENCES Autospecificaties(Id);

/\*\* 5 \*\*/

INSERT INTO AutoSpecificaties (Id)

SELECT Voertuig.AutoSpecificatiesId

FROM Voertuig

RIGHT JOIN AutoSpecificaties ON Voertuig.AutoSpecificatiesId = AutoSpecificaties.Id

GO

/\*\* 6 \*\*/

ALTER TABLE voertuig

ALTER COLUMN autospecificatiesid int NOT NULL;

/\*\* 7 \*\*/

ALTER TABLE voertuig

DROP COLUMN Merk, Model, Uitvoering, Catalogusprijs, kleur

/\*\* 8 \*\*/

CREATE VIEW Voertuigspecificatiescheck2 AS

SELECT Voertuig.AutoSpecificatiesId, AutoSpecificaties.Id

FROM Voertuig

LEFT JOIN AutoSpecificaties ON Voertuig.AutoSpecificatiesId = AutoSpecificaties.Id

GO

Select \* FROM Voertuigspecificatiescheck2