B6 Advanced MySQL

2020-2021

**Kevin Koster**

**Software Developer**



**Drenthe College Assen**

**Inhoud**

[Transactions - Antwoorden 3](#_Toc55995885)

[Stored procedures – Antwoorden 3](#_Toc55995886)

[Triggers & Events - Antwoorden 4](#_Toc55995887)

[Triggers & Events – Opdrachten 6](#_Toc55995888)

[Opdracht 1. 6](#_Toc55995889)

[Opdracht 2. 6](#_Toc55995890)

[Opdracht 3. 7](#_Toc55995891)

[Stored functions – Antwoorden 7](#_Toc55995892)

[Stored functions – Opdrachten 7](#_Toc55995893)

[Opdracht 1. 7](#_Toc55995894)

[Opdracht 2. 8](#_Toc55995895)

[Views – Antwoorden 8](#_Toc55995896)

[Views – opdracht 8](#_Toc55995897)

[Eindopdrachten 9](#_Toc55995898)

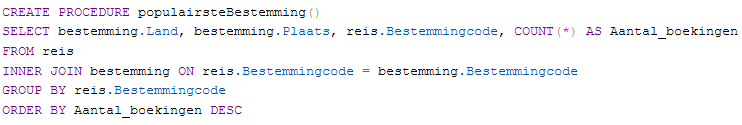
[A. 9](#_Toc55995899)

[B. 9](#_Toc55995900)

## Transactions - Antwoorden

1. Een transactie is één of een reeks opdrachten die in logische volgorde uitgevoerd word op de database.
2. Bij een online betaling kan het handig zijn, gezien het niet de bedoeling is dat als er iets fout gaat, dat jij dan wel geld betaald maar geen product krijgt.
3. Een *COMMIT* is het opslaan van het resultaat van één of meerdere uitgevoerde query’s. Een *ROLLBACK* daarentegen is het tegenovergestelde, namelijk: het terugdraaien van de database naar de status hoe het was voordat de transactie begon om zo het resultaat ongedaan te maken.
4. Een *MySQL-SAVEPOINT* is een punt in een transactie waar je een *ROLLBACK* naar kunt doen. Elke transactie die na zo’n *SAVEPOINT* gedaan word, die word teruggedraaid. In tegenstelling tot *ROLLBACK*, die dus alle wijzigingen terugdraait vanaf het begin van de transactie.

## Stored procedures – Antwoorden

1. Een stored procedure kun je zien als een functie die query’s uitvoert op de database.
2. Je zou bijvoorbeeld na het uitvoeren van een query kunnen kijken of de data goed ingevoerd is in de database.
3. Ik denk dat het nodig is om de juiste boeking te krijgen, vervolgens heb je de juiste reis nodig bij deze boeking. Om de plaats, het land en het werelddeel te zien heb je de juiste bestemming nodig voor de reis.
4. Dit is hoe je een *stored procedure* maakt buiten een transaction:   
     
   en dit is hoe je deze dan binnen een transaction aanroept:  
   

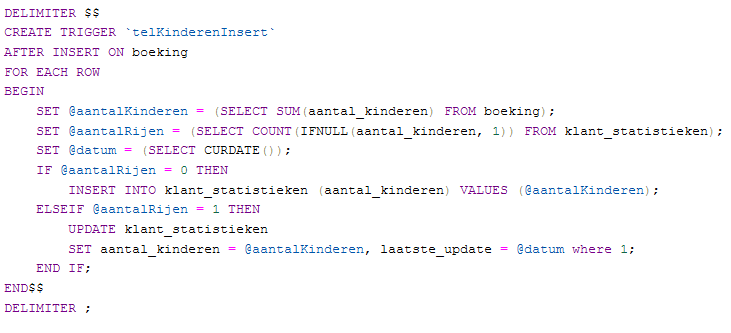
## Triggers & Events - Antwoorden

1. Als we het hebben over *‘telVolwassenenInsert’* dan word er na een insert opgeteld hoeveel volwassenen er in de *boeking* tabel zitten. Vervolgens word er gekeken of er al *rows* bestaan in de tabel *klanten\_*statistieken. Als er *rows* bestaan dan word het aantal volwassenen geupdate in de *klanten\_statistieken* tabel, zo niet dan word er een *insert* gedaan in de tabel.
2. Er word een *insert* gedaan van het totaal aantaal volwassenen.  
   
3. Er word een *update* gedaan van het totaal aantal volwassenen en *laatste\_update* word aangepast.  
   
4. Er word weer een *update* gedaan van het totaal aantal volwassenen. Volgens de code word *laatste\_update* ook nogmaals geupdate, maar dit is niet te zien omdat ik het op dezelfde dag gedaan heb. 
5. *DELIMITER* veranderd het teken voor het aangeven van het einde van een statement. Dit houdt in dat als ik “DELIMITER ;” gebruik, dat ik met een ‘;’ het einde van een statement aan geef.
6. *telVolwassenenInsert*:1. **Na een *insert*** op de tabel *boeking* vragen we het totaal aantal volwassenen op uit de tabel *boeking.  
   2.* Vervolgens vragen we het aantal *rows* op van *aantal\_volwassenen* in *klant\_statistieken*.   
   3. De datum word opgevraagd door *CURDATE().   
   4.* Als het aantal *rows* 0 is, dan *inserten* we de @aantalVolwassenen variabele als *aantal­\_volwassenen* in *klant\_statistieken*.  
   5. Als het aantal *rows* 1 is dan *updaten* we het aantal volwassenen met de variabele @aantalVolwassenen.  
     
     
     
     
     
     
     
   *telVolwassenenDelete:*1. **Na een *delete***op de tabel *boeking* vragen we het totaal aantal volwassenen op uit de tabel *boeking*.  
   *2.* Vervolgens vragen we het aantal *rows* op van *aantal\_volwassenen* in *klant\_statistieken*.   
   3. De datum word opgevraagd door *CURDATE().   
   4.* Als het aantal *rows* 0 is, dan *inserten* we de @aantalVolwassenen variabele als *aantal­\_volwassenen* in *klant\_statistieken*.  
   5. Als het aantal *rows* 1 is dan *updaten* we het aantal volwassenen met de variabele @aantalVolwassenen.  
     
   *telVolwassenenUpdate:*1. **Na een *update***op de tabel *boeking* vragen we het totaal aantal volwassenen op uit de tabel *boeking*.  
   *2.* Vervolgens vragen we het aantal *rows* op van *aantal\_volwassenen* in *klant\_statistieken*.   
   3. De datum word opgevraagd door *CURDATE().   
   4.* Als het aantal *rows* 0 is, dan *inserten* we de @aantalVolwassenen variabele als *aantal­\_volwassenen* in *klant\_statistieken*.  
   5. Als het aantal *rows* 1 is dan *updaten* we het aantal volwassenen met de variabele @aantalVolwassenen.
7. Zoals in het vorige voorbeeld gedaan is, om bijvoorbeeld statistieken bij te houden.
8. Een trigger is een *stored procedure* die uitgevoerd word wanneer een *event* plaatsvindt. Dus bijvoorbeeld als er een insert gedaan word op de database tabel.

## Triggers & Events – Opdrachten

### Opdracht 1.

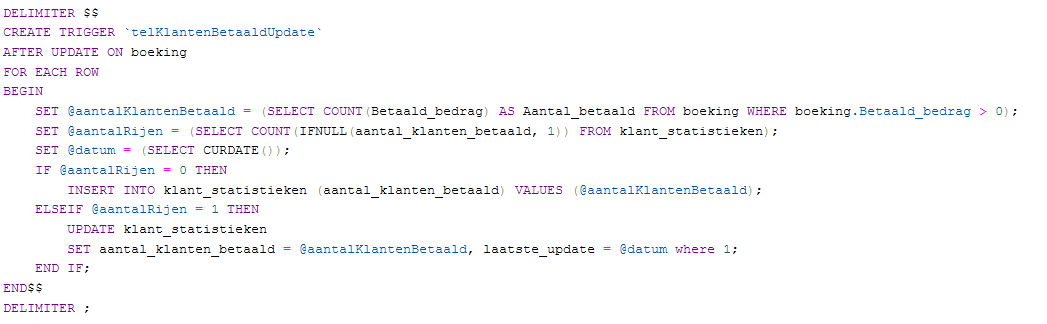
Maak voor de kolom Aantal\_kinderen 1 ‘INSERT’ trigger & event aan.





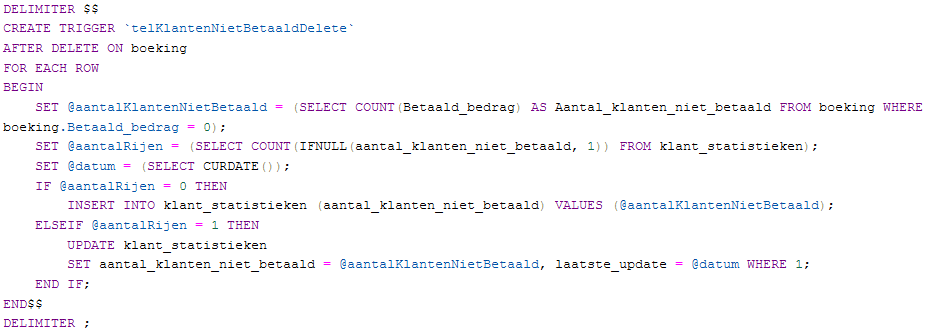
### Opdracht 2.

Maak voor de kolom Aantal\_klanten\_betaald 1 ‘UPDATE’ trigger & event aan.


### Opdracht 3.

Maak voor de kolom Aantal\_klanten\_betaald 1 ‘DELETE’ trigger & event aan.





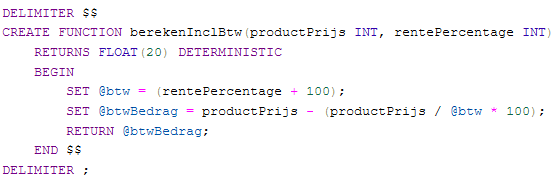
## Stored functions – Antwoorden

1. *DETERMINISTIC* geeft altijd hetzelfde resultaat voor dezelfde input parameters,   
   *NOT DETERMENISTIC* geeft verschillende resultaten voor dezelfde input parameters.
2. Het geeft aan wat voor type waarde de functie als resultaat heeft.
3. Een stored function geeft **altijd** een waarde terug, bij een stored procedure is dit optioneel.

## Stored functions – Opdrachten

### Opdracht 1.

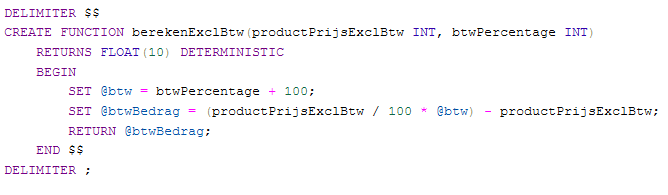
In deze opdracht gaan we een functie maken waarbij we de btw van een product inclusief btw moeten gaan berekenen. De terugkerende waarde zou dan het bedrag van de btw moeten zijn.





### Opdracht 2.

We gaan nou een functie aanmaken die een bepaald btw-percentage over een product zonder btw moet gaan berekenen. De terugkerende waarde zou dan het bedrag van de btw moeten zijn.

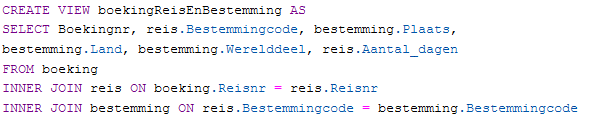


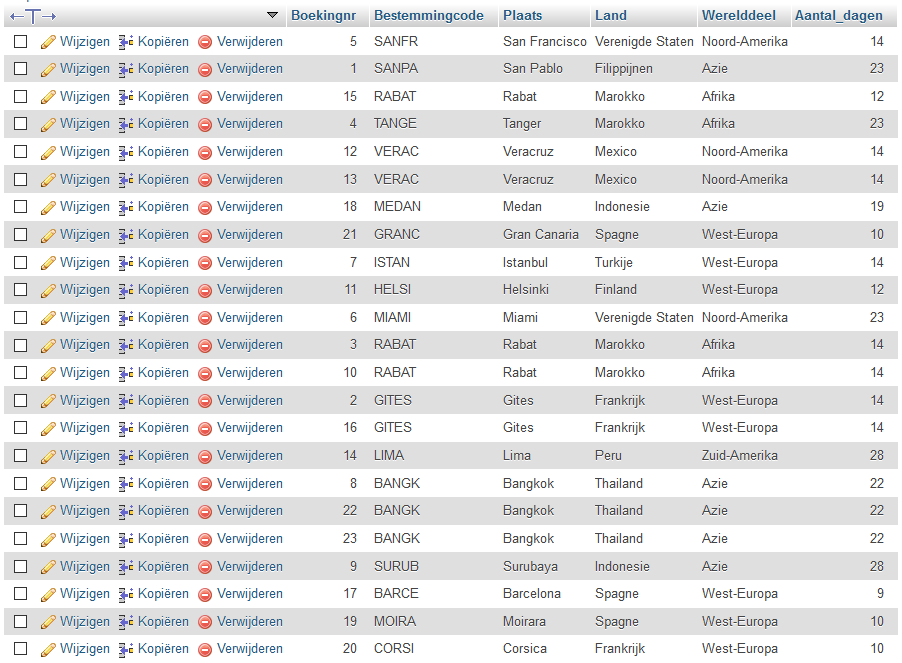


## Views – Antwoorden

1. Een overzicht van één of meerdere tabellen.
2. Een virtuele tabel maken met daarin de juiste rijen en kolommen.
3. Ja

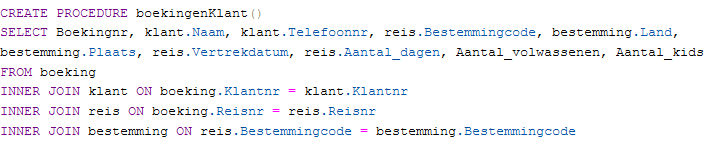
## Views – opdracht

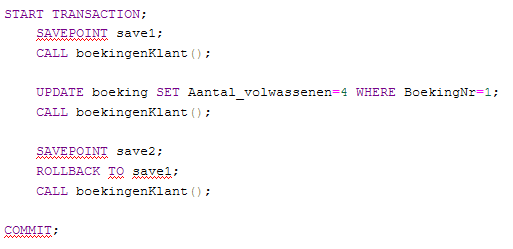




## Eindopdrachten

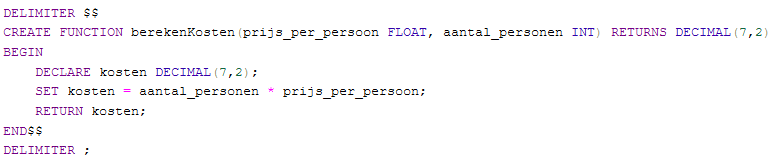
### A.

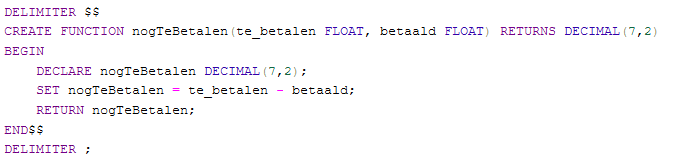


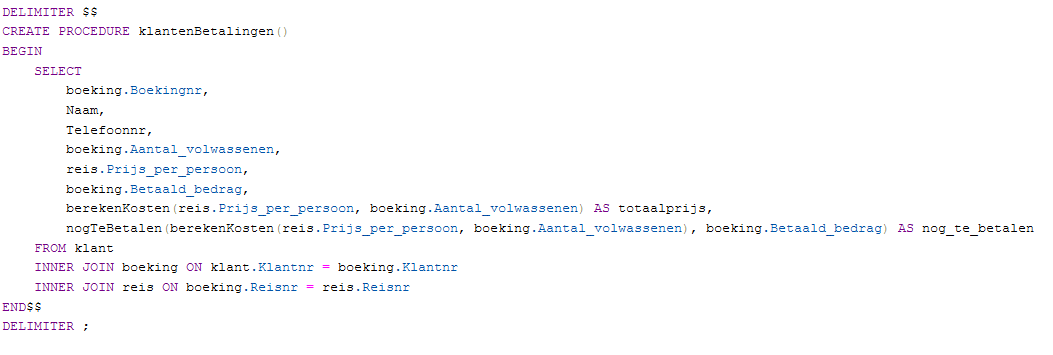


In deze *transaction* roep ik eerst *boekingenKlant()* aan om te zien welke waarde hier eerst stond voor Aantal\_volwassenen, vervolgens doe ik een update en roep ik weer de procedure aan om te zien of dit gelukt is. Daarna doe ik een rollback naar het eerste SAVEPOINT zodat alle wijzigingen ongedaan gemaakt worden. Nogmaals roep ik de procedure aan om te laten zien dat alle wijzigingen weer ongedaan gemaakt zijn.

### B.







Met deze procedure vraag ik een overzicht op van wie er hoeveel betaald heeft en wat er nog betaald moet worden. Ik bereken de totaalprijs van de reis door de prijs per persoon keer het aantal volwassenen te doen. Vervolgens gebruik ik *nogTeBetalen()* om te berekenen wat er nog betaald moet worden.