**Transaction Vragen**1. Een transactie is eigenlijk een process waar het een ‘alles of niets’ mentaliteit heeft. Dat betekent dat er iets gebeurd (bijvoorbeeld je gaat geld pinnen), maar er gaat wat fout waardoor het geld niet uit de machine komt. Nu zorgt de transactie ervoor dat het geld dan ook niet uit het account gaat.   
  
2. Een transactie zou kunnen gebruiken worden bij een parkeerautomaat.   
  
3. Een commit is een wijziging in code die werkt en die je op wilt slaan. Voor wanneer het fout gaat, dat je altijd terug kan gaan naar die laatste commit met een rollback.  
  
4. Een savepoint kun je zien als een checkpoint in een transactie. Stel je voor er gaat na de savepoint iets fout, hoef je niet helemaal terug, maar kun je terug gaan naar de savepoint en bespaart dus tijd.

**Stored Procedure Vragen**1. Een stored procedure is een mySQL query die opgeslagen en terug geroepen kan worden met een variabele. Bijvoorbeeld een lijstje laten zien van elke reis waar iedereen heen is geweest.  
2. Je kunt het gebruiken voor diverse lijstjes en om informatie van een database in en makkelijke tabel aan te schouwen.  
3. Er zijn zoveel *inner joins* omdat het zoekt in meerdere tabellen. Het zoekt voor gelijkheden tussen de tabellen. De Query zoekt bijvoorbeeld naar gelijken tussen de reis en bestemming tabel met de Bestemmingscode  
4.  
*Het aanmaken van de procedure*  
CREATE PROCEDURE boekingAantaldatum()

SELECT Boekdatum, klant.Naam, reis.Vertrekdatum, reis.Aantal\_dagen FROM boeking

INNER JOIN klant ON boeking.Klantnr = klant.Klantnr

INNER JOIN reis ON boeking.Reisnr = reis.Reisnr

*De Transactie*START TRANSACTION;

CALL boekingAantaldatum;

ROLLBACK;

**Trigger & Events Vragen** 1.  
-De trigger begint pas nadat er gegevens in de table *boeking* zijn toegevoegd.   
-Het begint met het tellen van alle volwassenen in de *boeking* table.   
-Daarna telt het de aantal rijen in *klant\_statistieken* (inclusief de rijen die een NULL waarde hebben)  
-Daarna wordt de datum opgeslagen met de datum die er nu is.  
-Als het aantal rijen 0 is; Voeg de aantal volwassenen toe in de *klant\_statistieken* table  
-Als er 1 rij is update het programma de table met de laatste gegevens.  
  
2. Die table krijgt een update van het aantal\_volwassenen. Dat blijft optellen  
3. De table wordt alsnog bijgewerkt. De aantal\_volwassenen ging (in mijn geval) omhoog.  
4. Het aantal\_volwassenen field is omlaag gegaan.  
5. De delimiter zorgt ervoor dat wanneer je functies, procedures en triggers defineert dat mysql weet waar de procedure/functie/trigger eindigt en voert deze uit als één unit in plaats van meerdere individuele statements  
6.  
Trigger & event met delete statement  
Nadat er een rij wordt verwijderd in *boeking* telt de event weer alle volwassenen op uit de *boeking* table. Daarna kijkt de event naar het aantal rijen in *klant\_statistieken*, en set de datum met de datum die er die dag is. Als het aantal rijen in *klant\_statistieken* 0 is insert hij het aantal volwassenen in de *klant\_statistiek* table. Is er al een rij wordt de *aantal\_volwassenen* *en laatste\_update* field geüpdate met de bijbehorende informatie  
  
Trigger & event met update statement  
Nadat er een rij wordt geüpdate in *boeking* telt de event weer alle volwassenen op uit de *boeking* table. Daarna kijkt de event naar het aantal rijen in *klant\_statistieken*, en set de datum met de datum die er die dag is. Als het aantal rijen in *klant\_statistieken* 0 is insert hij het aantal volwassenen in de *klant\_statistiek* table. Is er al een rij wordt de *aantal\_volwassenen* *en laatste\_update* field geüpdate met de bijbehorende informatie  
  
7. Je zou het kunnen gebruiken voor een opsomming van een field. Op Trello gebruikte ik een trigger die een card automatisch aan een checklist toevoegde, en als de card op done werd gezet dat de checklist automatisch bijgewerkt werd.  
  
8. Een trigger is eigenlijk een sein dat wordt gegeven wanneer iets gebeurd. Je kan zelf zeggen in welke instantie dat seintje gegeven moet worden. De event luistert als het ware naar het seintje, en doet een specifieke taak wanneer hij het seintje hoort.

**Trigger & Events Opdrachten**1.

BEGIN

SET @aantalKinderen = (SELECT SUM(Aantal\_kids) FROM boeking);

SET @aantalRijen = (SELECT COUNT(IFNULL(Aantal\_kinderen, 1)) FROM klant\_statistieken);

SET @datum = (SELECT CURDATE());

IF @aantalRijen = 0 THEN

INSERT INTO klant\_statistieken (Aantal\_kinderen) VALUES (@aantalKinderen);

ELSEIF @aantalRijen = 1 THEN

UPDATE klant\_statistieken

SET Aantal\_kinderen = @aantalKinderen, Laatste\_update = @datum WHERE 1;

END IF;

END  
  
2.

BEGIN

SET @aantalBetaald = (SELECT COUNT(Betaald\_Bedrag) FROM boeking WHERE Betaald\_bedrag > 0);

SET @aantalRijen = (SELECT COUNT(IFNULL(Aantal\_Klanten\_Betaald, 1)) FROM klant\_statistieken);

SET @datum = (SELECT CURDATE());

IF @aantalRijen = 0 THEN

INSERT INTO klant\_statistieken (Aantal\_Klanten\_Betaald) VALUES (@aantalBetaald);

ELSEIF @aantalRijen = 1 THEN

UPDATE klant\_statistieken

SET Aantal\_Klanten\_Betaald = @aantalBetaald, Laatste\_update = @datum WHERE 1;

END IF;

END

3. BEGIN

SET @aantalNietBetaald = (SELECT COUNT(Betaald\_Bedrag) FROM boeking WHERE Betaald\_bedrag = 0);

SET @aantalRijen = (SELECT COUNT(IFNULL(aantal\_klanten\_niet\_betaald, 1)) FROM klant\_statistieken);

SET @datum = (SELECT CURDATE());

IF @aantalRijen = 0 THEN

INSERT INTO klant\_statistieken (aantal\_klanten\_niet\_betaald) VALUES (@aantalBetaald);

ELSEIF @aantalRijen = 1 THEN

UPDATE klant\_statistieken

SET aantal\_klanten\_niet\_betaald = @aantalNietBetaald, Laatste\_update = @datum WHERE 1;

END IF;

END