Blok 6 - Databases / PDO

- Database 1
- SQL
- Extra uitleg SQL
- PDO
- CRUD - Challenge
- Kennis Check blok 6
- New Page

Database 1

1 Wat is een database?

datasource

? Leerdoelen

- Je weet wat een database is en waarvoor die gebruikt wordt.
- Je kunt gegevens uit de echte wereld omzetten naar tabellen en kolommen.
- Je begrijpt het verschil tussen ruwe gegevens en een gestructureerd model.

? Uitleg

In het dagelijks leven slaan organisaties gegevens op: over klanten, producten, studenten, boeken, etc. Een **database** is een digitale plek waar zulke gegevens netjes georganiseerd worden bewaard. Je verdeelt de informatie over verschillende tabellen, waarbij elke tabel over één onderwerp gaat, zoals:

- **Studenten**: naam, klas, geboortedatum
- **Docenten**: naam, vak, afdeling
- **Opleidingen**: naam, niveau, duur

Een database lijkt een beetje op een Excel-bestand, maar is veel krachtiger en beter gestructureerd. Je wilt geen dubbele gegevens en alles moet logisch met elkaar verbonden zijn.

Voorbeeld: Dit is géén goede database:

```
Naam         | Klas         | Opleiding
------------ | ------------ | -----------------
Fatima       | SD1A         | Software Developer
Ali          | SD1A         | Software Developer
Robin        | SD2B         | Software Developer
Jente        | SD1A         | Software Developer
Steven       | SD1A         | Recht & HR
```

→ Hier zie je dat de opleiding meerdere keren herhaald wordt. Dat is zonde en foutgevoelig.

□□ Wat zou beter zijn?

Je zou in plaats daarvan een aparte tabel 'Studenten' maken en een aparte tabel 'Opleidingen'. Studenten krijgen dan een *verwijzing* naar hun opleiding (dat komt in de volgende lessen aan bod).

?? Opdracht 1 – Gegevens analyseren

1. Bekijk onderstaande lijst met gegevens:

```
Naam: Esra
Klas: SD1A
Opleiding: Software Developer
Docent: De Jong
Vak: Webontwikkeling
Lesdag: Woensdag
```

2. Welke verschillende **onderwerpen** zie je hierin? Probeer per onderwerp de eigenschappen op te schrijven.
   - **Student**: naam, klas
   - ... (jij vult aan)
3. Zet je antwoorden in een tabelvorm: welke tabellen zou je nodig hebben? Welke kolommen zouden erin staan?

? Reflectie

- Waarom is het vastleggen van gegevens op meerdere plaatsen foutgevoelig? Beschrijf een situatie waarin dat fout kan gaan.
- Waarom is het niet handig om alle informatie in één grote tabel te zetten?
- Wat denk je dat het voordeel is van losse tabellen met verbindingen?

? Inleveren

- Lever je tabellenindeling in (.txt, .pdf of screenshot).
- Voeg je antwoorden toe op de reflectievragen.

2 Entiteiten en Attributen

? Leerdoelen

- Je weet wat een entiteit is en wat een attribuut is.
- Je kunt entiteiten en hun attributen herkennen in een realistisch scenario.
- Je kunt een eerste versie van een ERD tekenen met entiteiten en attributen.

? Uitleg

◫ Entiteit

In de vorige opdracht heb je '**onderwerpen**' en '**eigenschappen**' bepaald, weet je het nog? Een onderwerp was 'student' en een eigenschap was 'naam' en 'klas'.

Bij het ontwerpen van een database heet een onderwerp een 'entiteit', en een eigenschap is een ' attribuut'

> Een **entiteit** is een "***ding***", "***persoon***" of "***gebeurtenis***" in de echte wereld waarover je gegevens wilt opslaan.

Bijvoorbeeld:

- `Student` -> persoon
- `Cursus` -> gebeurtenis
- `Docent` -> persoon
- `Opleiding` -> gebeurtenis
- `Laptop` -> ding

◫ Attribuut

Een **attribuut** is een eigenschap van een entiteit. Bijvoorbeeld:

- De entiteit `Student` heeft de attributen: `studentnummer` `voornaam`, `achternaam`, `telefoonnummer`.
- De entiteit `Cursus` heeft de attributen: `naam`, `duur`, `startdatum`.

□□ ERD

Een entiteit met de attributen zet je in een bepaald formaat in een database ontwerp. Dat heet een ERD.

Op de eerste regel staat de **naam van de entiteit** en in de middelste kolom zet je alle **attributen** .

Voorbeeld:

image.png

□□ In een ERD (Entity Relationship Diagram) teken je entiteiten als rechthoeken en attributen als ovale of gelabelde velden ernaast.

> In een ERD heb je **drie kolommen**
> de eerste voor de **keys**, de tweede voor de **attribuutnamen** en de derde voor de **datatypes**.

?? Opdracht 1 – Entiteiten en attributen

1. Je werkt voor een evenementenbureau. Zij willen bijhouden:
   - Welke klanten boekingen doen
   - Welke evenementen er zijn
   - Welke locaties beschikbaar zijn
2. Maak een lijstje van minstens 3 entiteiten uit deze situatie. Bijvoorbeeld:
   - `Klant`
   - ...
   - ...
3. Geef per entiteit minstens 3 bijpassende attributen. Bijv.:
   - Klant → voornaam, e-mailadres, ...
   - Locatie → naam, ...
4. Teken jouw eerste ERD in Lucidchart.

Lucichart

Registreer je voor Lucichart en maak een gratis account.

Kies toevoegen library

image.png

Zoek naar ERD en selecteer "Entity Relationship"

image.png

Gebruik vervolgens dit figuur om een entiteit te maken.

image.png

? Reflectie

- Hoe weet je of iets een entiteit was of gewoon een attribuut?
- Heb je misschien dingen dubbel in verschillende entiteiten? Kun je iets beter loskoppelen?

? Inleveren

- Maak je ERD in Lucichart en maak een screenshot.

3 Primary Keys

? Leerdoelen

- Je weet wat een **primary key** is.
- Je begrijpt waarom een primary key verplicht is in elke tabel.
- Je kunt per entiteit een geschikte primary key kiezen.

? Uitleg

□□ Wat is een Primary Key (PK)?

Een primary key is een uniek gegeven waarmee je één rij uit een tabel kunt identificeren. Elke tabel in een database **moet** een primary key hebben.

Regel: Elke entiteit heeft precies één PK. Heb je meerdere opties? Kies er één. Heb je geen goede kandidaat? Gebruik dan een kunstmatige sleutel, zoals een `id`.

Voorbeelden:

- `Student` → `student_nr`
- `Auto` -> `kenteken`
- `Evenement` → `evenement_id`
- `Klant` → email_adres

Een primary key moet:

- Uniek zijn
- Nooit leeg zijn
- Vast blijven (mag niet wijzigen)

Als je later tabellen met elkaar verbindt, gebruik je de PK om een verbinding te maken van de ene entiteit naar de andere entiteit.

□□ Voorbeeld van een ERD met PK

image.png

?? Opdracht 1 – Kies je primary keys

1. Gebruik het ERD van de vorige opdracht met de entiteiten: `evenement`, `klant` en `locatie`.
2. Voeg aan elke entiteit een geschikte primary key toe. Kies daarbij een bestaand aatribuut of voeg zelf een sleutel toe.
3. Zet in de eerste kolom van je ERD de letters PK om aan te geven dat deze regel de PK bevat.
4. Controleer: is jouw PK echt uniek en onveranderlijk?

? Reflectie

- Welke van je gekozen PK's is natuurlijk (bestaand gegeven), en welke kunstmatig (gegenereerd ID)?
- Wat zou er misgaan als je geen PK kiest?

? Inleveren

- Lever een screenshot van je ERD in met in de eerste kolom de PK's (primary keys).

4 1:N-relaties en Foreign Keys

? Leerdoelen

- Je weet wat een 1:N-relatie is in een database.
- Je begrijpt wat een foreign key (FK) is en waarvoor die dient.
- Je kunt zelf een 1:N-relatie modelleren met een foreign key.

? Uitleg

□□ Wat is een 1:N-relatie?

Bij een **één-op-veel-relatie** (1:N) hoort bij één rij in de ene tabel, meerdere rijen in de andere tabel.

Bijvoorbeeld:

- `1 Klant` → **`veel`** `Boeking(en)`
- `1 Docent` → **`veel`** `Cursussen`

De **“meer”-kant** krijgt de foreign key (FK). De FK is een kopie van de primary key (PK) van de andere tabel.

> □□ De regel: **Meer = harkje = FK**

- Teken het harkje aan de kant waar “veel” is
- In die tabel voeg je de foreign key toe
- Voorbeeld: `boekingen` krijgt `klant_id` als FK

□□ Voorbeeld

Een klant kan veel boekingen hebben (andersom kan niet!). Dus het harkje komt aan de kant van de boeking.

Bij elk harkje hoort een **FK** die verwijst naar de **PK** van de entiteit waarmee die is gekoppeld. In dit geval dus klantnummer.

image.png

◫ Het `klantnummer` in boeking is een *foreign key* die verwijst naar klant.

?? Opdracht 1 – Relaties en foreign keys

1. Gebruik het ERD van de vorige opdracht met de entiteiten: `evenement`, `klant` en `locatie`.
2. Bepaal de veel kant van klant en evenement (evenement zou een concert kunnen zijn). Je wilt bijhouden welke evenementen door een klant worden bezocht.
3. Bepaal de veel kant van evenement en locatie
4. Teken de entiteiten en de relaties. Zet de harkjes aan de juiste kant.
5. Voeg foreign keys toe (FK's) en plaats in de eerste kolom een FK bij elke Foreign key.

? Inleveren

- Lever je een screenshot van de ERD's inclusief foreign keys en erelaties in.

5 Datatypes en Validatie

? Leerdoelen

- Je kent de meest gebruikte datatypes in een database.
- Je kunt passende datatypes kiezen voor attributen in je ERD.
- Je weet waarom validatie belangrijk is bij het kiezen van datatypes.

? Uitleg

◫ Wat is een datatype?

Een datatype bepaalt welk soort informatie je in een kolom/tabel opslaat. Het zorgt ervoor dat je database weet hoe de data eruitziet en wat ermee mag gebeuren.

Niet alle data-typen zijn even snel als je ze in een database gebruikt. Als je één van de volgende veelgebruikte datatypes gebruikt dan zit je meestal goed. Wil je andere gebruiken zoek dan goed uit wat de nadelen zijn.

Veelgebruikte datatypes:

Datatype	Gebruik	Voorbeeld
`INT`	Voor hele getallen (bijv. ID's, aantallen)	`5`, `142`
`VARCHAR(50)`	Voor tekst tot X karakters	`'Jan'`, `'email@example.com'`
`DATE`	Voor datums zonder tijd	`'2025-06-06'`
`DATETIME`	Voor datum én tijd	`'2025-06-06 14:30:00'`
`DECIMAL(6,2)`	Voor getallen met komma (bijv. prijs)	`12.99`, `9999.00`

Voorbeeld

image.png

Tips voor goede keuzes:

- Let op lengte bij `VARCHAR`: hoe langer, hoe trager, maar te kort is ook niet goed.
- Een telefoonnummer is een `VARCHAR()` en geen `INT`, waarom?
- Stel je wil een bedrag van maximaal 9999,99 opslaan dan gebruik je decimal(6,2).

?? Opdracht 1 – Pas je datatypes aan

1. Gebruik het ERD van de vorige opdracht met de entiteiten: `evenement`, `klant` en `locatie`.
2. Voeg de volgende attributen toe:
   - evenement -> toegangsprijs, aanvangsdatum en aanvangstijd
3. Kies voor elk attribuut een passend datatype uit de tabel hierboven.
4. Schrijf de datatypes in de derde kolom van de ERD's..
5. Gebruik minimaal 3 verschillende soorten datatypes; je mag er zelf attributen bij bedenken.

? Inleveren

- Lever je bijgewerkte ERD in met daarin duidelijk per attribuut het datatype genoteerd en de FK.

6 Case – Modelleer een realistisch scenario

? Leerdoelen

- Je kunt zelfstandig entiteiten en relaties herkennen in een realistisch scenario.
- Je past de 5 stappen toe om een ERD te maken.
- Je maakt een logisch en technisch correct ERD met PK’s, FK’s en datatypes.

Herhaling, de 5 basisregels

1. Een **entiteit** is een persoon, ding of gebeurtenis. Een getal of bedrag (bijvoorbeeld gewicht) is nooit een entiteit, maar altijd een attribuut (=eigenschap) van een entiteit.
2. Elke entiteit heeft precies één **PK (primary key)**. De primary key maakt de entiteit uniek (bijvoorbeeld kenteken van een auto).
3. Entiteiten hebben de volgende **relaties** 1:1, 1:N, N:1 of N:M.
   - 1:1 relaties bestaan bijna niet, als ze voorkomen dan kun je de relaties samenvoegen.
   - 1:N en N:1 is eigenlijk hetzelfde en komen het meest voor.
4. Een **1:N** relatie verbind je met een lijntje met een 'harkje'. Het **lijntje** staat aan de 1-kant en het '**harkje**' staat aan de meer-kant.
5. Bij elk 'harkje' hoort precies één FK. De FK verwijst naar de PK van de table waarmee deze is verbonden.

? Uitleg

Tot nu toe heb je geleerd wat entiteiten, attributen, PK’s, FK’s, datatypes en 1:N-relaties zijn. Nu pas je alles toe op een echte situatie.

□□ Scenario: Fietsenmaker Snelle Jelle

Fietsenmaker Snelle Jelle wil na een **reparatiebeurt** zijn klanten per SMS of Whatspapp op de hoogte stellen dat de reparatie klaar is. In dit bericht wil hij ook vertellen hoe hoog de reparatiekosten zijn.

Omdat de veel **klanten** meer dan één **fiets** hebben, wil hij van de fietsen ook wat kenmerken vastleggen. Hij wil het merk, model, type en kleur kunnen vastleggen.

Van elke reparatiebeurt wil hij verder vastleggen wanneer het onderhoud plaatsvond, hoe lang de reparatie duurde, wat er is uitgevoerd en de prijs.

□□ Je hebt de volgende entiteiten

- **reparatie**: ....
- **klant**: ....
- **fiets**: ....

?? Opdracht – Maak het ERD

1. Bepaal per entiteit eerst alle attributen, lees daarvoor goed het scenario door!
2. Teken de drie entiteiten in Lucichart
3. Zet alle attributen in de entiteiten
4. Bepaal de Primary Key (PK).
5. Bepaal de data-types.
6. Bedenk wat de relaties zijn en teken die met het hartje aan de goede kant.
7. Bepaal de Foreign Key (FK)

? Inleveren

- Lever een screenshot in van je ERD gemaakt in Lucichart. Zorg dat alles goed leesbaar is.

7 Meerdere relaties en N:N

? Leerdoelen

- Je begrijpt wat een N:N-relatie is en wanneer die voorkomt.
- Je kunt een N:N-relatie correct omzetten naar aparte tabellen met FK’s.
- Je kunt meerdere relaties per entiteit modelleren in een ERD.

? Uitleg

□□ Wat is een N:N-relatie?

Een **veel-op-veel** (N:N) relatie komt voor wanneer meerdere records uit entiteit A gekoppeld kunnen zijn aan meerdere records uit entiteit B.

Voorbeeld: Studenten kunnen zich inschrijven voor meerdere vakken. Elk vak kan meerdere studenten hebben.

□□ Hoe modelleer je dit?

Je maakt een extra tabel tussen de twee entiteiten. Deze bevat alleen de foreign keys van beide kanten.

image.png

De tussentabel (ook koppeltabel genoemd) bevat meestal ook extra informatie, zoals de inschrijfdatum of het cijfer. Deze extra informatie gaat over de combinatie student-vak. Een student heeft geen cijfer, een vak heeft geen cijfer, maar de combinatie student-vak heeft wel een cijfer.

□□ Meerdere relaties op één entiteit?

Soms is een entiteit aan meerdere andere entiteiten verbonden.

Voorbeeld: Een **medewerker** werkt in een afdeling, maar ook aan meerdere projecten.

- 1 medewerker ↔ 1 afdeling → 1:N
- 1 medewerker ↔ meerdere projecten ↔ N:N

Gebruik verschillende relaties als het logisch is dat een entiteit meerdere rollen vervult.

?? Opdracht – N:N-model

1. Maak een ERD voor dit scenario:
   Een muziekschool organiseert **lessen**. **Leerlingen** kunnen zich inschrijven op meerdere

lessen. Elke **les** wordt gevolgd door meerdere leerlingen.

2. Bedenk zelf voor elke entiteit minimaal 4 attributen.
3. Modelleer de juiste entiteiten, attributen, PK’s, FK’s en datatypes.
4. Voeg een tussentabel toe om de N:N-relatie correct te verwerken.

? Reflectie

- Waarom is een tussentabel nodig bij N:N?
- Welke extra informatie kun je in de tussentabel kwijt?
- Waar moet je op letten bij het toevoegen van meerdere relaties in je ERD?

? Inleveren

- Lever een screenshot in van je ERD gemaakt in Lucichart. Zorg dat alles goed leesbaar is.
- Beantwoord de reflectievragen (pdf of txt bestand)

8 Bibliotheek

□□ Scenario: Bibliotheek

In een bibliotheek wil men bijhouden welke klanten welk boek van welke periode tot periode hebben geleend.

Elke klant kan meerdere boeken gelijktijdig lenen. Verder wil men de klant een whatsapp kunnen sturen twee dagen voor het verstrijken van de inleverdatum.

- Maak een databaseontwerp (ERD).

? Inleveren

- Lever een screenshot in van je ERD gemaakt in Lucichart. Zorg dat alles goed leesbaar is.

--

Aanpassen 2026, het voorbeeld klant booking locatie is eigenlijk een N:M relatie.
- neem ander voorbeeld: klant - order - product
- vermeld de 5 stappen om een ERD te maken duidelijker, maak die structuur helderde.

SQL

1 Introductie phpMyAdmin en SELECT

datasource

? Leerdoelen

- Je weet hoe je XAMPP en phpMyAdmin gebruikt.
- Je kunt een database importeren.
- Je weet wat SQL is.
- Je kunt gegevens ophalen met `SELECT` en `FROM`.

? Uitleg

In deze eerste opdracht werk je met een database met informatie over films. Je gebruikt de `SELECT` -instructie om gegevens op te halen uit de tabel `movies`.

Je voert verschillende queries uit om de inhoud van de tabel te verkennen.

Wat is SQL precies?

SQL staat voor **Structured Query Language**. Het is een programmeertaal die speciaal is ontworpen om te "praten" met databases. Een database is eigenlijk gewoon een heel goed georganiseerde verzameling van gegevens. Denk bijvoorbeeld aan de databases van:

- **Instagram:** met alle gebruikers, foto's, likes en reacties.
- **Fortnite:** met alle spelers, hun skins, V-Bucks en statistieken.
- **Een webshop:** met alle producten, prijzen en voorraad.

Met **SQL** kun je deze databases vragen stellen (queries) of opdrachten geven.

?Video - SQL introductie

https://www.youtube.com/watch?v=zpnHsWOy0RY

https://www.youtube.com/embed/zpnHsWOy0RY

Wat heb je nodig?

- XAMPP (Apache en MySQL moeten aan staan)
- De database `mdb_movies.sqlsql`

XAMPP en phpMyAdmin

We gaan gebruik maken van phpMyAdmin.

phpMyAdmin is eenonderdeel van XAMPP en wordt veel gebruikt om met datbases te werken. Je kan databases aanmaken, verwijderen, aanpassen en inzien.

Je kunt phpMyAdmin pas opstarten als je XAMPP goed draait; Apache en mySQL staan aan.

> image.png

Start localhost/phpmyadmin

> phpmyadmin-ui-1-1.jpg

Stappen om de database te importeren:

1. Start Apache en MySQL via het XAMPP Control Panel.
2. Ga naar `http://localhost/phpmyadmin` in je browser.
3. Maak een nieuwe database aan met de naam `imdb_movies`.
4. Selecteer de database en gebruik het tabblad *Import* om het bestand `imdb_movies.sql` te importeren.

Kom je er niet uit: hier staat met plaatjes uitgelegd hoe je een database importeert.

?? Opdracht

1. Je hebt de database imdb_movies.sql geimporteerd
2. Voer de volgende drie SQL-query’s uit op de tabel `movies`:

- `SELECT * FROM movies;`
- `SELECT movie FROM movies;`
- `SELECT movie, rating FROM movies;`

  Heb je meer uitleg nodig over hoe je query maakt, hier staat een voorbeeld met plaatjes)

3. Bekijk het resultaat van elke query. Wat valt je op?

? Reflectie

- Welke kolommen lijken jou het nuttigst als je een lijst met filmaanbevelingen zou maken?
- Wat is het verschil tussen `SELECT *` en `SELECT kolomnaam`?

? Inleveren

1. Maak een screenshot waarbij je laat zien dat je een query hebt uitgevoerd op de database imdb_movies.
2. Beantwoord de refelctievragen en lever die in (txt of pdf).

2 WHERE en logica

? Leerdoelen

- Je kunt gegevens filteren met `WHERE`.
- Je begrijpt het gebruik van logische operatoren zoals `=`, `>`, `<`, `AND`, en `OR`.
- Je kunt AI gebruiken om een query te genereren en deze zelf controleren en verbeteren.

? Uitleg

Met `SELECT` haal je gegevens op. Met `WHERE` kun je die gegevens filteren. Bijvoorbeeld: alleen landen met een hoge geluksindex, of alleen landen uit Europa.

We gebruiken de database mod-mysql-basic-worldhappiness.sql. Deze bevat o.a. tabellen `jaar2015` en `jaar2016` met kolommen als `country`, `region`, `rank` en `score`.

Voorbeelden:

```
SELECT * FROM jaar2016 WHERE score > 7000;
SELECT country, region FROM jaar2015 WHERE region = "Western Europe";
SELECT country, score FROM jaar2016 WHERE region = "Western Europe" AND score > 7300;
```

?? Opdracht

1. Importeer de database mod-mysql-basic-worldhappiness.sql in phpMyAdmin en selecteer de database `worldhappiness`.
2. Voer de volgende query's uit en controleer het resultaat:
   - Selecteer alle landen uit de tabel `jaar2015`.
   - Selecteer alleen `country` en `score` uit `jaar2016`.
   - Selecteer alle regio's uit 2015.
   - Selecteer alle scores hoger dan 7200 in 2016.
   - Selecteer landen uit de regio "Western Europe" in 2015.
3. Gebruik ChatGPT om een extra query te genereren waarbij je gebruik maakt van `OR`. Maak hierbij gebruik van de database `worldhappiness`

? Reflectie

- Welke query vond je het lastigst en waarom?
- Welke filters heb je gebruikt? Noem minstens twee logische operatoren.
- Wat heeft AI (ChatGPT) goed gedaan, en wat moest je zelf aanpassen?

? Inleveren

- Lever de een screenshot van de zelf gemaakte query (opdracht, stap 3) in, laat zien dat die werkt en leg uit in eigen woorden hoe die werkt.

3 Aggregatiefuncties

? Leerdoelen

- Je kunt gebruik maken van SQL-functies zoals `COUNT()`, `AVG()`, `SUM()`, `MIN()` en `MAX()`.
- Je kunt kolommen hernoemen met `AS` (alias).
- Je begrijpt het verschil tussen `SELECT` van rijen en het samenvatten van gegevens.

? Uitleg

Aggregatiefuncties worden gebruikt om samenvattingen te maken van gegevens in een kolom. Ze voeren een berekening uit op meerdere rijen tegelijk in plaats van één rij.

Belangrijkste functies:

Functie	Doel	Voorbeeld
`COUNT()`	Telt hoeveel rijen er zijn	`SELECT COUNT(*) FROM players;`
`AVG()`	Geeft het gemiddelde van een kolom met getallen	`SELECT AVG(wage) FROM players;`
`SUM()`	Telt alle waarden in een kolom bij elkaar op	`SELECT SUM(value) FROM players;`
`MIN()`	Laat de kleinste waarde zien	`SELECT MIN(age) FROM players;`
`MAX()`	Laat de grootste waarde zien	`SELECT MAX(value) FROM players;`

Alias gebruiken met `AS`

Je kunt je resultaatkolom een duidelijke naam geven met het sleutelwoord `AS`.

```
SELECT AVG(wage) AS gemiddeld_loon FROM players;
```

Bonus: afronden met `ROUND()`

```
SELECT ROUND(AVG(wage)) AS gemiddeld_loon_afgerond FROM players;
```

?? Opdracht

Gebruik de database mod-mysql-basic-fifa2018.sql

1. Voer de volgende queries uit in phpMyAdmin:
   - Toon het gemiddelde loon van alle spelers bij Ajax.
   - Toon de totale waarde van spelers onder de 20 jaar.
   - Toon het hoogste loon van een speler bij FC Utrecht.
   - Toon het aantal spelers uit Nederland.
   - Toon het gemiddelde loon van alle Braziliaanse spelers, afgerond op hele euro's.
2. Gebruik bij elke query een duidelijke **alias** via `AS`.

? Reflectie

- Wat is het voordeel van een samenvattende query (zoals `AVG()`) in plaats van het handmatig bekijken van individuele rijen?
- Welke query vond je het lastigst en waarom?

? Inleveren

- Lever een .txt-bestand in met alle 5 query's én de antwoorden.

Vergeet de aliasen niet!

4 DELETE en veiligheid

? Leerdoelen

- Je begrijpt het doel van een `DELETE`-statement.
- Je kunt een `DELETE`-statement schrijven met een `WHERE`-clausule.
- Je weet waarom een `WHERE`-clausule cruciaal is bij verwijderen van gegevens.

? Uitleg

Met SQL kun je niet alleen gegevens opvragen, maar ook verwijderen. Dat doe je met het `DELETE`-statement. Hierbij is het essentieel dat je altijd een `WHERE`-clausule gebruikt. Als je dat niet doet, verwijder je álle rijen in de tabel!

Voorbeeld:

```
DELETE FROM players WHERE name = "K. Huntelaar";
```

Deze query verwijdert alleen de speler met die naam.

Fout voorbeeld (NIET DOEN!):

```
DELETE FROM players;
```

Deze query verwijdert **alle spelers** uit de tabel. Dit is onherstelbaar.

?? Opdracht

Gebruik de database mod-mysql-basic-fifa2018.sql (zoals in de vorige opdracht).

1. Voer de volgende opdrachten uit met een `DELETE`-statement:
   - Verwijder de speler "David Silva".
   - Verwijder alle spelers van de club "Willem II".
   - Verwijder alle Braziliaanse spelers die ouder zijn dan 34.
   - Verwijder alle spelers waarvan het loon (=wage) hoger is dan 200000.
2. Gebruik daarna de volgende query om te controleren hoeveel spelers er nog zijn:

   ```
   SELECT COUNT(*) FROM players;
   ```

   Het antwoord dat hier uit moet komen als je alles goed hebt gedaan is **533**

   image.png

> TIP: maak eerst een select en als je de juiste resultaten terug krijgt vervang dan de **select *** in **delete**

Mocht het niet goed gaan dan kun je de database altijd opnieuw importeren en opnieuw beginnen.

? Reflectie

- Wat had er fout kunnen gaan als je de `WHERE`-clausule was vergeten?

- Waarom is het handig om altijd eerst een `SELECT` met dezelfde `WHERE`-clausule te doen voordat je `DELETE` uitvoert?

? Inleveren

- Voeg een screenshot toe van de output van je `SELECT COUNT(*)`-query.
- antwoord op de reflectievragen (txt of pdf)

5 Introductie tot JOINs

? Leerdoelen

- Je begrijpt het nut van tabellen koppelen met een `JOIN`.
- Je weet wat een `PRIMARY KEY` en `FOREIGN KEY` zijn.
- Je kunt een eenvoudige `INNER JOIN` uitvoeren.

? Uitleg

In een echte database worden gegevens vaak verdeeld over meerdere tabellen. Je gebruikt een **JOIN** om die tabellen aan elkaar te koppelen. Zo kun je bijvoorbeeld zien in welke klas een student zit en wie zijn studieloopbaanbegeleider is.

Wat is een `JOIN`?

Een JOIN combineert rijen uit twee tabellen op basis van een kolom die ze gemeenschappelijk hebben. Dit is vaak een id-veld zoals `klas_id`.

Voorbeeld:

Stel: we hebben twee tabellen:

```
studenten
+----+----------+-----------+----------+
| id | voornaam | achternaam| klas_id  |
+----+----------+-----------+----------+
| 1  | Fatima   | Bakker    | 101      |
| 2  | Noah     | de Vries  | 101      |
```

```
| 3  | Aziz     | Bouali    | 102      |


klassen
+------+----------+------------------+
| id   | klas_naam| studie_coach     |
+------+----------+------------------+
| 101  | SD1A     | meneer Willems   |
| 102  | SD1B     | mevrouw Jansen   |
```

Vraag: "*wie is de studiecoach van Fatima*?"
Antwoord: "*Meneer Willems*".
Klopt dat?

Met een JOIN kun je voor elke student zien in welke klas hij/zij zit én wie zijn of haar studiecoach is:

```
SELECT studenten.voornaam, studenten.achternaam, klassen.klas_naam, klassen.studie_coach
FROM studenten
INNER JOIN klassen ON studenten.klas_id = klassen.id;
```

De tabel **studenten** wordt verbonden met **klassen** waarbij de **primary key** (id) van **klassen** wordt verbonden met de **foreign key** (klas_id) van **studenten**

?? Opdracht

Maak een nieuwe database aan

- Open phpMyAdmin en maak een database aan met de naam `join_oefening`.

Maak deze twee tabellen aan

Maak een database en voer importeer deze database

```
CREATE TABLE studenten (
  id INT AUTO_INCREMENT PRIMARY KEY,
  voornaam VARCHAR(50),
  achternaam VARCHAR(50),
  klas_id INT
);

CREATE TABLE klassen (
```

```
  id INT AUTO_INCREMENT PRIMARY KEY,
  klas_naam VARCHAR(50),
  aantal_leerlingen INT,
  studie_coach VARCHAR(100)
);
INSERT INTO klassen (klas_naam, aantal_leerlingen, studie_coach) VALUES
('SD1A', 24, 'meneer Willems'),
('SD1B', 22, 'mevrouw Jansen'),
('SD1C', 25, 'meneer El Idrissi');

INSERT INTO studenten (voornaam, achternaam, klas_id) VALUES
('Fatima', 'Bakker', 1),
('Noah', 'de Vries', 1),
('Aziz', 'Bouali', 2),
('Eva', 'Peeters', 2),
('Liam', 'Meijer', 1),
('Sophie', 'van der Laan', 2),
('Daan', 'Mulder', 3),
('Aisha', 'Khan', 3),
('Javi','Alvarez',4);
```

Schrijf een JOIN-query

1. Voer deze query uit:

   ```
   SELECT studenten.voornaam, studenten.achternaam, klassen.klas_naam,
   klassen.studie_coach
   FROM studenten
   INNER JOIN klassen ON studenten.klas_id = klassen.id;
   ```

2. Pas de query aan zodat je alleen studenten uit klas `SD1A` toont.

AI - ChatGPT

Als we alle studenten laten zien, dan zien we 8 studenten. In de database staan 9 studenten en als we goed kijken dan wordt (de laatste) student "Javi Alvarez" niet getoond. Hoe komt dat?

- Vraag aan ChatGPT hoe dit zit (tip: het heeft met de soort join te maken)
- Vraag aan ChatGPT om een JOIN-query te schrijven die alle studenten toont, dus ook Javi Alvarez

- Test of de AI-query werkt. Verbeter hem indien nodig.

? Reflectie

- Leg uit waarom Javi Alvarez eerst niet werd getoond
- Wat heb je gedaag om Javi Alvarez wel te tonen, leg uit.
- Beschrijf het verschil tussen de twee soorten joins die je hebt gebruikt.

? Inleveren

- Lever je reflectie in als .txt of .pdf, beschrijf hierin goed wat je hebt gedaan en beantwoord de refelctievragen in eigen woorden.
- Lever de Chat geschiedenis in: wat heb je precies gevraagd en wat was het antwoord (txt of pdf bestand).

6 AI en eigen query's

? Leerdoelen

- Je kunt zelfstandig SQL-vragen formuleren en uitvoeren.
- Je kunt AI gebruiken om een query te genereren, verbeteren en controleren.
- Je begrijpt hoe je informatie uit meerdere tabellen combineert.

? Uitleg

De database die we in deze opdracht gaat gebruiken heeft twee tabellen:

- `student` – bevat gegevens van studenten (zoals naam en inschrijfdatum)
- `progress` – bevat studieresultaten gekoppeld aan studenten

Verzin zelf een logische naam voro deze database en maak een lege database aan.

Importeer daarna de data met dit bestand: student-progress.sql

De tabellen zijn gekoppeld via `student_id`. Je kunt ze combineren met een `JOIN`.

Voorbeeld JOIN:

```
SELECT student.first_name, student.last_name, progress.subject_name, progress.grade_percentage
FROM student
JOIN progress ON student.student_id = progress.student_id;
```

?? Opdracht A

Gebruik AI / ChatGPT

1. Toon alle studenten met hun volledige naam en inschrijfdatum.
2. Toon alle vakken (`subject_name`) en behaalde percentages (`grade_percentage`) van student met voornaam "Aaron".
3. Toon per van start_jaar 2024 het hoogste cijfer dat is behaald voor C++. Gebruik `MAX()`
4. Toon alle studenten die in blok "276c8c" zitten én een cijfer boven de 90% hebben behaald.
   (er zouden 7 studenten moeten worden getoond)

? AI-opdracht B

Laat ChatGPT een query voor je schrijven die de volgende vraag beantwoordt (of verzin zelf een goede vraag):

- *Welk vak heeft gemiddeld over alle jaren de hoogste score?*

Test de gegenereerde query in phpMyAdmin. Als de query niet werkt, probeer dan samen met ChatGPT te achterhalen wat er fout gaat. Corrigeer en leg uit wat je hebt aangepast.

? AI-opdracht C

Kan je samen met AI de volgende output maken. In deze tabel staan per jaar de gemiddelden per vak en in de laatste kolom staat het gemiddelde over alle jaren.

image.png

? Reflectie

- Welke query vond je het moeilijkst om te maken en waarom?

- Beschrijf welk SQL commando je door de AI opdrachten hebt bijgeleerd en beschrijf in eigne woorden wat dit commando doet.?

? Inleveren

- Lever een .txt-bestand in met alle 7 query’s uit deze opdracht (Opdracht A, B en CI).
- Voeg een screenshot toe van het resultaat van de moeilijkste query.
- Lever je reflectie in als .txt of .pdf.

? SQL Terugblik en Samenvatting

? Leerdoelen

- Je kunt de belangrijkste SQL-onderdelen die je hebt geleerd opsommen en uitleggen.
- Je kunt per onderdeel een voorbeeldquery schrijven.
- Je herkent waar je zelf nog onzekerheden of fouten maakt.

?? Opdracht

Maak een overzicht waarin je de belangrijkste SQL-onderdelen samenvat die je dit blok hebt geleerd.

1. Voor elk SQL-onderdeel of keyword geef je:
   - De naam (bijv. `SELECT`)
   - Een korte beschrijving in je eigen woorden
   - Een kort voorbeeld (één regel SQL is genoeg)
2. Werk dit netjes uit in een tabel of lijst. Gebruik minstens 10 begrippen, zoals:
   - `SELECT`, `FROM`, `WHERE`, `JOIN`, `AVG()`, `GROUP BY`, `ORDER BY`, `DELETE`, `AS`, `COUNT()`
3. Gebruik de termen table, row en column (table, rij en colum).
4. Je mag AI gebruiken om voorbeelden te controleren, maar de uitleg moet in jouw eigen woorden zijn.

? Voorbeeld (fragment)

SQL-onderdeel	Omschrijving (in eigen woorden)	Voorbeeldquery
`SELECT`	Gebruik je om aan te geven welke **kolommen** je wilt zien.	`SELECT naam FROM studenten;`
`WHERE`	Gebruik je om **rijen** te filteren op een voorwaarde.	`SELECT * FROM progress WHERE grade_percentage > 80;`

? Inleveren

- Werk de opdracht uit, maak een PDF en lever die in.

Fatima bestaat niet -> Aaron

Extra uitleg SQL

Database maken

Start MariaDB (MySQL) in XAMPP en ga naar http://localhost/phpmyadmin

Je kunt ook op de admin in XAMPP drukken:

image-1676379334123.png

In phpmyadmin, maak je een nieuwe database en noem die **student**.

image-1676378685637.png

Download een importbestand (in dit voorbeeld student.sql) en importeer deze.

image-1676378813133.png

Select

Om gegevens uit de database te halen is een aparte taal bedacht. Dit heet SQL. Een SQL-programmaatje wordt een query genoemd en telt meestal maar een paar regel.

De meeste query's halen gegeven uit de database en de meest eenvoudige query haalt alle gegevens uit één entiteit. Stel, je wilt alle informatie van alle studenten zien, dan is dit de query:

```
SELECT * FROM student
```

> SELECT betekent; haal de gegevens op.
>
> * betekent *alle* kolommen
>
> FROM betekent dat er een tabelnaam (entiteitsnaam) volgt en *student* is dan de naam van de tabel.

Voer de query uit in *phpmyadmin* onder het tabje SQL.

> image-1676379711258.png

Zorg er voor dat je de juiste database hebt geselecteerd (hebt aangeklikt).

Je hebt nu een * in de query gebruikt. Dat betekent dat je alle velden laat zien. Stel dat je alleen de voornaam en het email adres wilt afdrukken dan kan je dat als volgt.

```
SELECT voornaam, email FROM student
```

Aliasen (AS)

Stel dat je nu de kolomnamen wilt veranderen dan kan je aliassen gebruiken. Stel de kolomnaam *voornaam* veranderen in *Naam*, en *email* in *eMail* dan kan dat als volgt.

```
SELECT voornaam as 'Naam', email as 'eMail' FROM student
```

WHERE

Stel je wilt alle gegevens van de student met de voornaam *Grazia* zien. Dat doe je door een WHERE te gebruiken.

De standaard query wordt als volgt opgebouwd

```
SELECT kolomnaam1, kolomnaam2, .... of * voor alles
FROM tabelnaam
WHERE de conditie
```

De query die alle gegeven van de student *Grazia* opzoekt wordt dan

```
SELECT * FROM `student`
WHERE voornaam='Grazia'
```

Let op dat *Grazia* een string is en daarom tussen '(singel quotes) moet worden gezet.

Wildcards (LIKE)

Een conditie kan ook een zogenaamde wildcard bevatten. Een wildcard is bijvoorbeeld alle namen die met een a beginnen. Daarvoor gebruik je in plaats van de = een like en je gebruikt bijvoorbeeld 'a%' om aan te geven dat de naam met een a moet beginnen. Of bijvoorbeeld '%t' om aan te geven dat de naam met een t moet eindigen.

OK, stel willen alle gegevens van alles studenten zien van wie de voornaam met een a begint.

```
SELECT * FROM student
WHERE voornaam LIKE 'a%'
```

Hoeveel studenten hebben een voornaam die met een a begint?

Nog een voorbeeld, stel willen alle gegevens van alles studenten met een email adres dat met *.com* eindigt. Hoe doe je dat?

```
SELECT * FROM student
WHERE email  like '%.com'
```

AND en OR

Stel we willen alle gegevens van alle studenten waarvan de voornaam met een a of een b of een c begint. Dan kunnen we met OR (net als in PHP) deze condities combineren.

```
SELECT * FROM student
where voornaam like 'a%' OR voornaam like 'b%' OR voornaam like 'c%'
```

Stel dat je in deze bovenstaande query per ongeluk een AND had gebruikt, wat zou er dan gebeuren? Als je wilt kan je het uitproberen!

Stel we willen alle gegevens van alles studenten van wie de voornaam met een a begint én van wie het e-mailadres op *.com* eindigt.

Daarvoor hebben we een AND nodig om de twee condities te combineren.

```
SELECT * FROM student
where voornaam like 'a%' AND email like '%.com'
```

Nu een lastige. Stel we willen de studenten van wie de voornaam met een a, b of c begint en van wie het e-mailadres op *.com* eindigt. We moeten nu de AND en OR gaan combineren.

Als je dat doet dan is er een regel en dat is dat je de OR tussen haakjes moet zetten omdat die bij elkaar hoort.

Dus de gecombineerde query wordt dan;

```
SELECT * FROM student
where ( voornaam like 'a%' OR voornaam like 'b%' OR voornaam like 'c%' )
       AND email like '%.com'
```

PDO

1 Verbinden met een database

datasource

? Leerdoelen

- Je weet wat PDO is en waarom het gebruikt wordt.
- Je kunt verbinding maken met een database via PDO.
- Je begrijpt waarom het handig is om een apart bestand voor de connectie te maken.

? Uitleg

PDO (PHP Data Objects) is een moderne manier om met databases te werken in PHP. Het ondersteunt meerdere soorten databases zoals MySQL, SQLite en PostgreSQL, maar in deze lessen gebruiken we alleen MySQL.

image.png

Je maakt verbinding met een database via een zogeheten DSN (Data Source Name) en slaat de connectie op in een variabele. Om herhaling te voorkomen, zet je dit in een apart bestand zoals `connection.php`. Dit maakt het ook makkelijk om de instellingen aan te passen wanneer je de website van je localhost naar een liveserver verplaatst.

`connection.php`

Het bestand `connection.php` bevat de code om verbinding te maken met de database. In plaats van in elk PHP-bestand opnieuw een connectie te moeten schrijven, zet je die één keer netjes in dit aparte bestand. Zo houd je je code overzichtelijk en voorkom je fouten.

Door `connection.php` te gebruiken, hoef je later bij het online zetten van je site alleen in **dat ene bestand** de instellingen aan te passen (zoals wachtwoord of host), in plaats van in alle bestanden waar je met de database werkt.

```
<?php
$dsn = 'mysql:host=localhost;dbname=database_name;charset=utf8mb4';
$user = 'root';
```

```
$pass = '';

try {
  $pdo = new PDO($dsn, $user, $pass);
} catch (PDOException $e) {
  echo "Verbinding mislukt: " . $e->getMessage();
}
```

In het connection.php bestand wordt de database naam, en het user id en password ingesteld. Op een XAMPP ontwikkel server is standaard de user `root` en heeft geen password. Op een productieserver is dat natuurlijk ander!

? Opdracht

1. Maak een database `voorbeeld` aan met één tabel `dieren` met de kolommen `id` (INT, AUTO_INCREMENT, PRIMARY KEY), `naam` (VARCHAR), en `soort` (VARCHAR).
2. Maak een bestand `connection.php` dat de connectie maakt zoals hierboven.
3. Maak een tweede bestand `testverbinding.php` waarin je `require 'connection.php';` gebruikt om verbinding te maken.
4. Laat met `echo` zien of de verbinding is gelukt (bijv. "Verbinding gelukt!").

? Reflectie

- Welke manieren zijn er in PHP om met een database te werken, en waarom gebruiken wij PDO?
- Wat zijn voordelen van een apart `connection.php` bestand?
- connection.php heeft op een development omgeving een andere inhoud dan op een productieserver. Wat is het verschil en waarom?

? Inleveren

- Beantwoord in eigen woorden de reflectievragen (txt of pdf).

2 Gegevens uitlezen met - SELECT

? Leerdoelen

- Je weet hoe je gegevens uit een MySQL-database ophaalt met PDO.
- Je kunt een SELECT-query uitvoeren via PDO.
- Je kunt resultaten weergeven in HTML via PHP.

? Uitleg

In deze opdracht gebruik je een bestaande database met studentgegevens. Je voert met behulp van PDO een `SELECT`-query uit en toont de resultaten in een HTML-tabel.

Database

Je gebruikt een SQL-bestand student.sql om snel een database en tabel aan te maken met voorbeeldgegevens:

1. Open **phpMyAdmin** (via XAMPP of MAMP).
2. Klik op "Importeren".
3. Selecteer het bestand `student.sql` dat je van je docent krijgt of downloadt.
4. Klik op "Start" om het script uit te voeren. Je krijgt nu een database met de tabel `studenten`.

read.php

Maak `read.php` en zet daar deze code in:

```
<?php
require 'connection.php';

$sql = "SELECT id, voornaam, achternaam, woonplaats FROM studenten";
$stmt = $pdo->query($sql);
$studenten = $stmt->fetchAll();
?>
<table border="1">
  <tr>
    <th>ID</th>
    <th>Naam</th>
    <th>Woonplaats</th>
  </tr>
  <?php foreach ($studenten as $student): ?>
```

```
  <tr>
    <td><?= $student['id'] ?></td>
    <td><?= $student['voornaam'] . ' ' . $student['achternaam'] ?></td>
    <td><?= $student['woonplaats'] ?></td>
  </tr>
  <?php endforeach; ?>
</table>
```

?? Opdracht – studentenlijst weergeven

1. Importeer `student.sql` in phpMyAdmin om de database en tabel aan te maken.
2. Maak een bestand `read.php`.
3. Maak een `connection.php` bestand zoals dat in de vorige les is uitgelegd.
4. Gebruik de gegeven code en test of de lijst met studenten goed wordt weergegeven.
5. **Extra (otionele) opdracht:** Voeg de kolom `email` toe aan je SELECT-query en aan de HTML-tabel. Zorg ervoor dat het e-mailadres klikbaar is via een `mailto:` link.

? Reflectie

(zoals altijd: leg uit in eigen woorden!)

- Waar en hoe wordt het `connection.php` bestand ingelezen in `read.php`?
- Wat doet `$pdo->query()` precies?
- Wat is het verschil tussen `fetch()` en `fetchAll()`?
- Hoe toon je data uit een array netjes in een HTML-tabel?
- Wat doet een `mailto:`-link precies?

? Inleveren

- Lever het bestand `read.php` in via Teams of Canvas.
- Beantwoord de reflectievragen in een .txt of .pdf bestand en lever die ook in.

3 Studentgegevens toevoegen - INSERT

? Leerdoelen

- Je weet hoe je data toevoegt aan een database met PDO.
- Je kunt een formulier maken en de ingevoerde gegevens veilig verwerken.
- Je begrijpt het gebruik van `prepare()` en `execute()` in PDO.

? Uitleg

Als je gegevens naar de database wilt sturen (bijvoorbeeld via een formulier), gebruik je een **INSERT-query**. Bij PDO doe je dit veilig met `prepare()` en `execute()`. Zo voorkom je problemen zoals **SQL-injectie**. SQL-injectie kan worden gebruikt om te 'hacken' en wordt later in een ander module uitgelegd.

Je gebruikt `prepare()` om de query voor te bereiden met placeholders (`:naam`), en daarna geef je met `execute()` de daadwerkelijke waarden door.

Voorbeeld – voorbereiding op INSERT

Maak een bestand `create.php` met deze HTML en PHP-code:

```
<?php
require 'connection.php';

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
  $sql = "INSERT INTO studenten (voornaam, achternaam, woonplaats, email)
          VALUES (:voornaam, :achternaam, :woonplaats, :email)";
  $stmt = $pdo->prepare($sql);
  // Vul hier de juiste execute() aan
}
?>

<form method="post">
  <label>Voornaam: <input type="text" name="voornaam"></label><br>
  <label>Achternaam: <input type="text" name="achternaam"></label><br>
```

```
  <label>Woonplaats: <input type="text" name="woonplaats"></label><br>
  <label>E-mail: <input type="email" name="email"></label><br>
  <button type="submit">Toevoegen</button>
</form>
```

?? Opdracht – student toevoegen via formulier

1. Maak het bestand `create.php` aan en zet de code hierboven erin.
2. Test of het formulier zichtbaar is in je browser.
3. **Vul de ontbrekende code aan:** zorg dat de `execute()` functie de juiste data gebruikt uit het formulier.
4. Voeg een `echo` toe na het invoegen (bijv. “Student toegevoegd!”) zodat je weet dat het gelukt is.
5. Controleer in **phpMyAdmin** of de student correct is toegevoegd aan de database.

? Reflectie

(zoals altijd: leg uit in eigen woorden!)

- Waarom gebruik je `prepare()` in plaats van direct een query uitvoeren?
- Hoe weet PHP welke waarden in de query moeten komen?
- Wat is de functie van regel 4 (de regel die met if begint)?

? Inleveren

- Lever het bestand `create.php` in via Teams of Canvas.
- Beantwoord de reflectievragen in een .txt of .pdf bestand en lever die ook in.

4 Studentgegevens bewerken - UPDATE

? Leerdoelen

- Je weet hoe je bestaande data wijzigt in een database met PDO.
- Je kunt een formulier maken dat bestaande gegevens toont en laat aanpassen.
- Je begrijpt hoe je een `UPDATE`-query uitvoert met `prepare()` en `execute()`.

? Uitleg

Met een `UPDATE`-query kun je bestaande gegevens in de database aanpassen. Je gebruikt ook hier `prepare()` en `execute()` zodat de invoer veilig verwerkt wordt.

Vaak haal je eerst de huidige gegevens op, zodat de gebruiker weet wat hij gaat bewerken. Daarna verwerk je de aangepaste gegevens.

Voorbeeld – basisopzet update.php

Maak een bestand `update.php` en vul deze code in:

```
<?php
require 'connection.php';

$id = $_GET['id'] ?? null;

if (!$id) {
  echo "Geen ID opgegeven.";
  exit;
}

$stmt = $pdo->prepare("SELECT * FROM studenten WHERE id = ?");
$stmt->execute([$id]);
$student = $stmt->fetch();

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
  $sql = "UPDATE studenten SET voornaam = :voornaam, achternaam = :achternaam, woonplaats =
:woonplaats, email = :email WHERE id = :id";
  $stmt = $pdo->prepare($sql);
  // Vul hier de juiste execute() functie aan
}
?>

<form method="post">
```

```
  <label>Voornaam: <input type="text" name="voornaam" value="<?= $student['voornaam'] ?>"></label><br>
  <label>Achternaam: <input type="text" name="achternaam" value="<?= $student['achternaam'] ?>"></label><br>
  <label>Woonplaats: <input type="text" name="woonplaats" value="<?= $student['woonplaats'] ?>"></label><br>
  <label>E-mail: <input type="email" name="email" value="<?= $student['email'] ?>"></label><br>
  <button type="submit">Opslaan</button>
</form>
```

?? Opdracht – studentgegevens aanpassen

1. Maak een bestand `update.php`.
2. Gebruik de code hierboven. Let op: gebruik een bestaand ID uit je database in de URL (bijv. `update.php?id=3`).
3. Test of de juiste gegevens zichtbaar zijn in het formulier.
4. **Vul zelf de `execute()` regel aan** zodat de gegevens uit het formulier worden opgeslagen in de database.
5. Voeg na het opslaan een `echo` toe met "Wijziging opgeslagen".
6. Controleer of de database wordt aangepast.

? Reflectie

- Hoe weet PHP welk studentrecord moet worden aangepast, leg uit heo dat werkt?
- Waarom gebruik je een `WHERE` clausule in een `UPDATE` query?
- Hoe weet je vanuit de code **zeker** dat de juiste gegevens zijn opgeslagen?

? Inleveren

- Lever het bestand `update.php` in via Teams of Canvas.
- Beantwoord de reflectievragen in een .txt of .pdf bestand en lever die ook in.

5 Student verwijderen

? Leerdoelen

- Je weet hoe je een record uit een database verwijdert met PDO.
- Je begrijpt waarom je altijd moet controleren welk record je verwijdert.
- Je kunt veilig en gecontroleerd een `DELETE`-query uitvoeren.

? Uitleg

Met een `DELETE`-query verwijder je een record uit de database. Omdat deze actie niet teruggedraaid kan worden, wil je dit altijd eerst bevestigen. Ook gebruik je een `WHERE` clausule om precies aan te geven welk record je bedoelt.

In deze opdracht toon je eerst de naam van de student die je gaat verwijderen, zodat je als gebruiker weet wat je doet. Daarna kun je pas bevestigen en verwijderen.

Voorbeeld – delete.php

```
<?php
require 'connection.php';

$id = $_GET['id'] ?? null;

if (!$id) {
  echo "Geen ID opgegeven.";
  exit;
}

// TO DO: Haal hier de student op via SELECT zodat je zijn/haar naam kunt tonen
// $student = ...

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
  $stmt = $pdo->prepare("DELETE FROM studenten WHERE id = :id");
  $stmt->execute(['id' => $id]);
  echo "Student verwijderd.";
  exit;
}
?>

<p>Weet je zeker dat je student <strong><?= '... hier de naam' ?></strong> wilt
```

```
verwijderen?</p>

<form method="post">
  <button type="submit">Ja, verwijder</button>
</form>

<p><a href="read.php">Annuleer</a></p>
```

?? Opdracht – student verwijderen met naam

1. Maak een bestand `delete.php`.
2. Gebruik de bovenstaande code.
3. **Vul zelf de `SELECT`-query aan** zodat je de gegevens van de student ophaalt op basis van het ID.
4. Laat de voornaam en achternaam van de student zien in de bevestigingsvraag.
5. Test de pagina via de URL: `delete.php?id=3` (of een ander bestaand ID).

? Reflectie

- Waarom is het belangrijk om eerst de naam van de student te tonen voordat je verwijdert?
- Hoe weet PHP welke gegevens bij het opgegeven ID horen?
- Wat gebeurt er als je probeert een niet-bestaand ID te verwijderen?
- Waarom gebruiken we `POST` voor het verwijderen?

? Inleveren

- Lever het bestand `delete.php` in via Teams of Canvas.
- Beantwoord de reflectievragen in een .txt of .pdf bestand en lever die ook in.

6 Studenten zoeken

? Leerdoelen

- Je weet hoe je een `WHERE` clausule gebruikt in een SELECT-query met PDO.
- Je begrijpt hoe `LIKE` gebruikt wordt voor zoeken op (deel van) tekst.
- Je kunt input van een formulier gebruiken om gegevens te filteren.

? Uitleg

Met een `WHERE`-clausule kun je filteren welke rijen je uit de database haalt. Als je wilt zoeken op een deel van een naam, gebruik je `LIKE` met het procentteken `%`.

Bijvoorbeeld: `SELECT * FROM studenten WHERE voornaam LIKE :zoek`, en dan geef je `%zoekwoord%` mee aan de placeholder `:zoek`.

Voorbeeldopzet – zoek.php

Maak een bestand `zoek.php` met deze opzet:

```
<form method="get">
  <input type="text" name="zoek" placeholder="Zoek op voornaam">
  <button type="submit">Zoeken</button>
</form>

<?php
require 'connection.php';

$zoekwoord = $_GET['zoek'] ?? '';

if ($zoekwoord) {
  // TO DO: vul de juiste SELECT-query en fetchAll() hier aan
}
?>
```

?? Opdracht – zoekfunctie bouwen

1. Maak het bestand `zoek.php`.
2. Gebruik de bovenstaande code als startpunt.
3. **Vul zelf de SELECT-query aan** met een `WHERE voornaam LIKE :zoek` clausule.
4. Voer de query uit met `prepare()` en `execute()` – gebruik `%` wildcards om ook op deelwoorden te zoeken.

5. Toon de resultaten in een tabel met voornaam, achternaam, woonplaats en e-mail.

? Reflectie

- Wat is het verschil tussen een gewone vergelijking (`=`) en `LIKE`?
- Hoe werkt `%` in een LIKE-query precies?
- Wat gebeurt er als je een lege zoekterm invoert?
- Waarom gebruik je `prepare()` ook bij zo'n zoekopdracht?

? Inleveren

- Lever het bestand `zoek.php` in via Teams of Canvas.
- Beantwoord de reflectievragen in een .txt of .pdf bestand en lever die ook in.

7 Mini-project: CRUD applicatie

? Leerdoelen

- Je kunt een complete CRUD-toepassing bouwen met PDO.
- Je past SELECT, INSERT, UPDATE, DELETE en LIKE correct toe.
- Je gebruikt `connection.php` om je project onderhoudbaar te houden.

Wat is CRUD?

CRUD staat voor **Create, Read, Update, Delete** – dit zijn de vier basisbewerkingen die je met gegevens in een database kunt uitvoeren:

- **Create**: nieuwe gegevens toevoegen (bijv. een nieuwe student inschrijven)
- **Read**: gegevens opvragen en tonen (bijv. een lijst van alle studenten)
- **Update**: bestaande gegevens aanpassen (bijv. een fout corrigeren in een naam)
- **Delete**: gegevens verwijderen (bijv. een student uitschrijven)

Een CRUD-applicatie is een programma dat deze vier functies ondersteunt. In webontwikkeling gebruik je vaak formulieren en SQL in combinatie met PHP of andere programmeertalen om deze

acties uit te voeren.

? Uitleg

Je hebt nu alle onderdelen geleerd om een volledige webapplicatie te maken die met een database werkt. In dit project bouw je een kleine CRUD-app voor studenten waarin je:

- een lijst van studenten toont (**read**)
- nieuwe studenten kunt toevoegen (**create**)
- bestaande studenten kunt bewerken (**update**)
- studenten kunt verwijderen (met bevestiging) (**delete**)
- kunt zoeken op voornaam

?? Opdracht – CRUD-app bouwen

1. Maak een eigen map of project met minimaal de volgende bestanden:
   - `read.php` – toont de lijst van studenten
   - `create.php` – formulier om nieuwe studenten toe te voegen
   - `update.php` – formulier om bestaande studenten te bewerken
   - `delete.php` – bevestiging en verwijdering van een student
   - `zoek.php` – zoekfunctie op voornaam
   - `connection.php` – je databaseverbinding
2. Zorg dat je hiervan één web applicatie van en zorg dat je via een menu bar kan navigeren.
3. Voeg bovenaan `read.php` een navigatie toe zodat je snel naar de andere pagina’s kunt.
4. Controleer of je code werkt voor verschillende studenten en test het met minstens 3 zelf ingevoerde records.
5. Je mag de opmaak aanpassen met CSS als je wilt, maar dat is optioneel.

? Reflectie

- Welke onderdelen van PDO vind je makkelijk, en welke lastig?
- Waar moet je vooral op letten bij UPDATE en DELETE?

- Wat zou je verbeteren als je meer tijd had voor dit project?
- Wat betekent het als we zeggen dat PDO "veilig" is? Wanneer is dat waar?

? Inleveren

- Lever je hele projectmap in (alle .php-bestanden).
- Lever de reflectie in (txt of pdf). En gebruik je eigen woorden!

8 Toetsvragen: PDO en CRUD

? Leerdoelen

- Je herkent de belangrijkste onderdelen van PDO-gebruik in PHP.
- Je begrijpt hoe CRUD-operaties technisch en logisch werken.
- Je kunt fouten of onvolledigheden in PDO-code herkennen en verklaren.

?? Toetsvragen

Wat doet PDO, waarvoor hebben we het nodig?

PDO staat voor PHP Data Objects. Het is een library met functies waarmee je PHP kan 'verbinden' met een database. Je kan dan in PHP SQL queries gebruiken om de data in de database op te vragen of aan te passen.

Hoe maak je een verbinding met de database via PDO?

Je maakt een DSN-string aan (bijv. `mysql:host=localhost;dbname=voorbeeld;charset=utf8mb4`), geeft gebruikersnaam en wachtwoord door, en instancieert een nieuwe `PDO`-object.

Het meest eenvoudige voorbeeld is:

```
$pdo = new PDO( 'mysql:host=localhost;dbname=database_name;charset=utf8mb4','user', 'password
');
```

Waarom is het handig om connection.php te gebruiken?

Door de connectiecode in één bestand te zetten, onderhoud je die centrale plek makkelijker.

Als je wachtwoord wijzigt of je plaatst je code vanaf je laptop op een productie server dan hoef je op maar één paats je wachtwoord aan te passen.

Waarom zou je trouwens je wachtwoord willen aanpassen als je je code op een prodctie server zet?

Op jouw laptop hgebruik je voor je database meestal geen password, lekker makkelijk. Een productieserver kan de hele wereld 'zien' en als je geen password gebruikt dan is je database binnen de kortste keren gehacked.

Tip: verander niet alleen je wachtwoord, maar pas ook je **user name** aan dan maak je het hackers nog iets lastiger.

Hoe gebruik je fetchAll() en wanneer gebruik je fetch()?

`fetchAll()` haalt alle rijen in één keer op als array van arrays (of objects). `fetch()` haalt per aanroep één rij op, handig als je maar één record verwacht of in een loop wilt verwerken.

Waarom gebruik je bij INSERT en UPDATE altijd POST in plaats van GET?

Omdat `POST` bedoeld is voor acties die data wijzigen en niet in de URL verschijnt, zodat gevoelige data niet zichtbaar wordt en de actie niet per ongeluk herhaald wordt bij refresh van de pagina.

Waarvoor gebruik je WHERE bij een UPDATE of DELETE?

De `WHERE`-clausule specifieert precies welke rij(en) aangepast of verwijderd moeten worden. Zonder `WHERE` wordt de operatie op álle rijen uitgevoerd.

Wat betekent CRUD en hoe komen deze onderdelen terug in de module?

CRUD staat voor Create, Read, Update, Delete. In de module leer je:

- **Create:** via `INSERT` en `prepare()/execute()` (create.php).
- **Read:** via `SELECT` en `fetchAll()` (read.php).
- **Update:** via `UPDATE` met prepared statements (update.php).
- **Delete:** via `DELETE` met bevestiging en `WHERE` (delete.php).

Wat doet require 'connection.php' ?

Dit voegt het bestand `connection.php` in en maakt de databaseconnectie beschikbaar. Hierdoor hoef je niet in elk stukje code waarin je PDO gebruikt opnieuw de connectie te schrijven.

? Inleveren

Gebruik deze vragen om jezelf te testen of als voorbereiding op de kennis-check.

Maak één zelf bedachte vraag over PDO gaat en die hier boven niet staat.

- Stel de vraag, geef het antwoord en een korte uitleg. Gebruik het formaat dat hier ook wordt gebruikt, Lever in in PDF of TXT.

--

CRUD - Challenge

8 CRUD Challenge – Te laat meldingen

datasource

? Leerdoelen

- Je kunt een volledige CRUD-toepassing bouwen met PDO en PHP.
- Je past invoercontrole en gebruikersinteractie toe in formulieren.
- Je begrijpt de rol van `prepare()`, `execute()`, en veilige query's.

? Uitleg

In deze challenge bouw je een complete toepassing waarin te laat meldingen van studenten worden bijgehouden. Je gebruikt alles wat je geleerd hebt over forms, PDO, databaseverwerking en CRUD-operaties.

Het eindresultaat is een overzichtspagina met alle meldingen, waarin je nieuwe meldingen kunt toevoegen, bestaande meldingen kunt wijzigen en meldingen kunt verwijderen (met bevestiging).

Voorbeeld

Bekijk het voorbeeld op: stampwerk.nl. Probeer de knop '*Weer eentje te laat*' en kijk wat er gebeurt.

De bedoeling is dat je een CRUD gaat maken. Wat is een CRUD? Dat leggen we zo uit. Eerste het voorbeeld, dat kan je vinden op:

https://stampwerk.nl

image.png

Wat is een CRUD?

CRUD staat voor Create, Read, Update en Delete.

image.png

Deze vier functies zijn de basisfuncties die je op een tabel uit de database kan uitvoeren. Stel je hebt een tabel student, je kunt een student toevoegen (Create), je kunt een overzicht krijgen van studenten (Read), je kunt de gegevens van een student aanpassen (Update) en als laatste kun je een student ook weer verwijderen.

De challenge is dat jij een CRUD gaat maken voor te laat meldingen. Maak daarvoor eerst een tabel in de database waarin je te laat meldingen kan registreren.

Als je naar het voorbeeld kijkt dan zie je dat je van een te laat melding de volgende gegevens wilt vastleggen:

- naam van de student
- klas
- aantal minuten te laat
- de reden van het te laat komen.

?? Stappenplan

Stap 1 – Database maken

- Maak een database en een tabel `meldingen` met de volgende velden:
  - `id` (INT, AUTO_INCREMENT, PRIMARY KEY)
  - `student` (VARCHAR)
  - `klas` (VARCHAR)
  - `minuten` (INT)
  - `reden` (TEXT)
- Voeg zelf enkele (minimaal 3) testregels toe via PHPMyAdmin.

Stap 2 – Read: overzicht maken

- Maak `read.php` waarin je alle meldingen toont in een HTML-tabel.
- Gebruik `require 'connection.php'` voor de databaseverbinding.
- Gebruik `query()` en `fetchAll()` om gegevens op te halen.
- Voeg **bovenaan** knoppen toe voor “toevoegen” en “zoeken”.

- "zoeken" staat er als knop maar hoef je niet uit te werken.

Stap 3 – Create: melding toevoegen

- Maak `create.php` met een formulier.
- Voeg invoervelden toe voor student, klas, minuten en reden.
- Sla de gegevens op met een `INSERT`-query via `prepare()` en `execute()`.
- Controleer invoer:
  - geen lege velden
  - minuten is een positief getal
- Toon een foutmelding bij **ongeldige** invoer en een **succesmelding** bij correcte invoer.
- *Front-end controle mag, en **back-end** controle moet en je moet kunnen laten zien dat het werkt.*

Stap 4 – Delete: melding verwijderen

- Voeg op `read.php` een knop “verwijder” toe per rij.
- Laat deze verwijzen naar `delete.php?id=...`.
- Toon eerst een bevestigingspagina met de gegevens van de melding.
- Voer pas na bevestiging de `DELETE`-query uit.

Stap 5 – Update

- Voeg op `read.php` een knop “wijzig” toe per rij.
- Laat deze verwijzen naar `update.php?id=...`.
- Controleer op de update pagina of het id is meeggegeven em of het id bestaat.
- *Ontbreekt het id of is deze niet aanwezig dan laat je een foutmelding zien.*
- Haal de bestaande gegevens op met een `SELECT`-query.
- Toon een formulier met ingevulde waarden.
- Werk de gegevens bij met een `UPDATE`-query.

? Reflectie

- Wat heb je geleerd over het werken met databases in PHP?
- Wat ging er goed, en waar had je hulp bij nodig?
- Wat zou je in een volgend project anders aanpakken?
- Hoe zorg je ervoor dat je database veilig blijft bij gebruikersinvoer?

? Inleveren

- Eén screenshot van je overzichtspagina (`read.php`).
- Eén screenshot van je invoerpagina (`create.php`).
- Eén screenshot van je wijzigpagina (`update.php`).
- Alle PHP-bestanden, SQL-export van de database, en eventuele CSS.
- Een reflectieverslag in .txt of .pdf met de bovenstaande vragen beantwoord.
- Maak een afspraak met een docent die je uitlegt hoe je code werkt.

? Puntentelling

Je moet minimaal **81 van de 100 punten** halen om deze opdracht succesvol af te ronden.

Punten	Onderdeel
10	**Database en tabel**: juiste velden, datatypes en primary key zijn aangemaakt.
10	**Read**: overzichtspagina toont meldingen correct in een tabel.
10	**Layout**: nette opmaak met CSS (Bootstrap, Tailwind, ...), duidelijke structuur. Gebruik het voorbeeld als uitgangspunt.
20	**Create**: formulier voegt een nieuwe melding toe, inclusief invoercontrole.
10	**Invoercontrole**: negatieve of ongeldige invoer wordt opgevangen met een melding.
10	**Delete**: verwijderen werkt inclusief bevestiging én juiste studentnaam.
20	**Update**: bestaand record kan worden aangepast via een formulier met ingevulde velden.
10	**Codekwaliteit & veiligheid**: gebruik van `prepare()`, `execute()`, nette bestandsstructuur.

Kennis Check blok 6

Kennis Check blok 6

datasource

1. Wat is een database en waarom gebruiken we die?

Een database is een digitale 'schatkist' waarin je gegevens over bijvoorbeeld studenten, producten of klanten netjes opslaat.

Je gebruikt een database omdat het overzichtelijk is, snel werkt en voorkomt dat je dingen dubbel bijhoudt.

2. Hoe zet je ‘echte wereld’ gegevens om naar tabellen en kolommen?

Je kijkt naar de onderwerpen (zoals 'Student', 'Opleiding') en noemt die entiteiten. Van elk onderwerp maak je kolommen (attributen), zoals naam, klas of geboortedatum. Zo maak je structuur in je data.

3. Wat is het verschil tussen ruwe gegevens en een gestructureerd model?

Ruwe gegevens zijn zoals je ze tegenkomt: alle info door elkaar. Een gestructureerd model brengt orde: je verdeelt info over tabellen met duidelijke kolommen. Daardoor raak je minder snel in de war.

4. Waarom is het foutgevoelig om gegevens meerdere keren op te slaan?

Stel dat je opleiding ‘Software Developer’ vijf keer typt en één keer tikfout maakt. Dan heb je verschillende versies van dezelfde opleiding – dat maakt fouten en verwarring. Daarom bewaar

je info op één plek.

5. Wat is een entiteit en wat is een attribuut?

Een entiteit is een 'dingen'-soort, zoals Student of Cursus. Attributen zijn eigenschappen daarvan, zoals naam, klas of duur. Zo kun je entiteiten en eigenschappen herkennen en verwerken.

6. Waarom hebben we altijd een primary key nodig?

Een primary key is een uniek kenmerk in een tabel, bijvoorbeeld student_id. Zo weet je zeker dat elke rij uniek is en kun je records makkelijk terugvinden en koppelen.

7. Wat is een 1:N-relatie en hoe maak je die in een ERD?

Bij een 1:N-relatie hoort één entiteit bij meerdere bij een andere. Bijvoorbeeld één klant -> meerdere boekingen. Je tekent een haakje aan de kant met “veel” en zet een foreign key in de ‘veel’-tabel.

8. Wat is een foreign key en waarom gebruiken we die?

Een foreign key is een verwijzing in de ene tabel naar de primary key van een andere tabel. Daarmee maak je een verband tussen twee tabellen, zoals boeken gekoppeld aan klanten.

9. Wat zijn datatypes en waarom moet je die zorgvuldig kiezen?

Datatypes bepalen wat voor soort gegevens je opslaat, zoals INT, VARCHAR of DATE. Als je het verkeerde datatype kiest, kan je database fout gaan of traag worden. En voor telefoon gebruik

je VARCHAR omdat het niet rekent.

10. Wat is een N:N-relatie en hoe los je dat op in je model?

Een N:N-relatie is een relatie waarin meerdere records van entiteit A gekoppeld zijn aan meerdere van B, bijvoorbeeld leerlingen met meerdere lessen. Dat los je op met een tussentabel waarin de foreign keys van beide entiteiten samen komen.

SQL

1. Waarom gebruik je verschillende SQL-commando's zoals SELECT, INSERT, UPDATE en DELETE?

Omdat je met elke opdracht iets anders doet met de gegevens. Met `SELECT` haal je gegevens op, met `INSERT` voeg je iets toe, met `UPDATE` verander je iets, en met `DELETE` verwijder je iets.

Als je maar één commando zou hebben, zou je alles zelf moeten bouwen. Nu is het veel makkelijker en duidelijker.

2. Wat gebeurt er als je een WHERE-clausule vergeet bij een UPDATE- of DELETE-opdracht?

Dan pas je alles aan of verwijder je alles in de hele tabel! Dat is meestal niet wat je wilt. De `WHERE` zorgt ervoor dat je alleen die rijen verandert die aan een bepaalde voorwaarde voldoen, bijvoorbeeld: alleen de student met ID 5.

3. Waarom gebruik je JOIN?

Stel: je hebt een tabel met studenten en een andere met klasnamen. Als je wilt zien in welke klas een student zit, moet je de tabellen combineren. Dat doe je met een `JOIN`. Dan kun je

gegevens uit beide tabellen tegelijk laten zien, alsof het één lijst is.

4. Stel: je hebt een tabel 'studenten' en je wilt alleen de studenten uit Amsterdam zien, welke SQL-structuur gebruik je?

Dan gebruik je `WHERE woonplaats = 'Amsterdam'` achter je SELECT.

Zo zeg je: "Geef alleen de rijen waarvan de woonplaats gelijk is aan Amsterdam."

5. Wat is het nut van een wildcard zoals '%' in een LIKE-zoekopdracht?

De `%` betekent: 'maakt niet uit wat hier staat'. Als je zoekt op `naam LIKE 'J%'`, krijg je alle namen die beginnen met een J, zoals "Jesse", "Julia" of "Joris". Het is handig als je niet precies weet wat iemand heeft ingevuld.

6. Waarom is het belangrijk om kolommen te kiezen in SELECT, in plaats van SELECT *?

Met `SELECT *` haal je alles op, ook dingen die je niet nodig hebt. Dat kan je website trager maken en is onduidelijk. Als je alleen de kolommen kiest die je nodig hebt, is je code netter en sneller.

7. Wat verandert er aan je data-structuur door een PRIMARY KEY toe te voegen?

Een `PRIMARY KEY` zorgt ervoor dat elke rij uniek is, bijvoorbeeld op ID.

De rij die je een `PRIMARY KEY` geeft, moet dus ook uniek zijn; van elke waarde mag er maar één voorkomen.

8. Hoe zou jij een fout in een query ontdekken en oplossen?

Lees goed wat de foutmelding zegt. Vaak is het een typefout, een komma die mist of een fout in de naam van een kolom. Probeer de query stap voor stap te testen. Gebruik bijvoorbeeld phpMyAdmin om te kijken of je query daar werkt.

Als je er niet uit komt dan kan je de query ook stapje voor stapje eenvoudiger maken, door telkens iets weg te laten.

9. Wat is het verschil tussen een SQL-query schrijven in phpMyAdmin en dezelfde query uitvoeren via PHP/PDO?

In phpMyAdmin voer je de query handmatig in – één keer. Met PHP/PDO doe je dat in je code, automatisch, telkens als iemand je website gebruikt. Zo kun je gegevens tonen of opslaan zonder handmatig iets te doen.

PDO

Waarom kiezen we voor PDO boven oudere methodes zoals mysql_* of mysqli?

PDO is veiliger en flexibeler. Het helpt je om makkelijker over te stappen naar een andere database (bijvoorbeeld van MySQL naar SQLite).

PDO is het meest recent en modern. PDO wordt verder ontwikkeld en mysqli `minder.

PDO is iets eenvoudiger in gebruikt.

Wat is een DSN in PDO?

DSN betekent "Data Source Name". Dat is een stukje tekst waarin staat met welke database je verbinding maakt.

Waarom is het handig om de verbinding (bijvoorbeeld connection.php) in een apart bestand te zetten?

Zo hoef je niet overal dezelfde code te typen. Als er iets verandert (zoals het wachtwoord), dan hoef je dat maar op één plek aan te passen. Je maakt je code netter, overzichtelijker en makkelijker te onderhouden.

Wat is het verschil tussen de database-instellingen op je eigen laptop en op een echte server?

Op je laptop gebruik je vaak "localhost" en een gebruiker (*root*) zonder wachtwoord.

Op een echte server is dat niet veilig, dus daar gebruik je andere gegevens. Het is belangrijk dat je die gegevens netjes gescheiden houdt, anders werkt je site straks niet online of wordt hij gehackt.

Wat is het verschil tussen query() en prepare() + execute()?

Met `query()` stuur je direct een SQL-opdracht.

Met `prepare()` + `execute()` gebruik je een veilige manier waarbij je eerst zegt "wat je ongeveer wilt doen" en daarna de gegevens pas invult. Dat is veiliger, vooral bij gegevens van een formulier.

Hoe helpt prepare() om SQL-injecties te voorkomen?

Een SQL-injectie is als iemand iets raars intypt in een formulier om je database kapot te maken. `prepare()` zorgt ervoor dat die invoer niet als SQL-code wordt gezien, maar gewoon als tekst. Daardoor kunnen hackers niks kapot maken.

Hoe haal je meerdere rijen uit een database met PDO?

Je doet eerst `$stmt = $conn->query("SELECT ...")`, dan gebruik je `fetch()` om één rij te krijgen of `fetchAll()` voor alles tegelijk.

`fetch()` is handig als je maar één resultaat verwacht, zoals bij een login. `fetchAll()` gebruik je als je bijvoorbeeld een lijst van producten wilt laten zien.

?? Opdracht

Maak de kennis-check.

? Inleveren

Aan het einde van de kennis-check ontvang je een certificaat. Maak een schermafdruk en lever deze in.

--

New Page

8 CRUD Challenge – Te laat meldingen

datasource

? Leerdoelen

- Je kunt een volledige CRUD-toepassing bouwen met PDO en PHP.
- Je past invoercontrole en gebruikersinteractie toe in formulieren.
- Je begrijpt en gebruikt `prepare()` en `execute()` voor veilige query's.

? Uitleg

In deze challenge bouw je een complete toepassing waarin te laat meldingen van studenten worden bijgehouden. Je gebruikt alles wat je hebt geleerd over formulieren, PDO, databaseverwerking en CRUD-operaties.

Het eindresultaat is een overzichtspagina met alle meldingen. Je kunt nieuwe meldingen toevoegen, bestaande meldingen aanpassen en meldingen verwijderen (met bevestiging).

Voorbeeld

Bekijk het voorbeeld op: https://stampwerk.nl. Probeer de knop '*Weer eentje te laat*' en kijk wat er gebeurt.

Wat is een CRUD?

CRUD staat voor Create, Read, Update en Delete.

Dit zijn de vier basisbewerkingen op een database:

- Create: gegevens toevoegen
- Read: gegevens ophalen en tonen
- Update: gegevens aanpassen

- Delete: gegevens verwijderen

In deze opdracht maak je een CRUD-toepassing voor te laat meldingen. Je slaat per melding de volgende gegevens op:

- naam van de student
- klas
- aantal minuten te laat
- reden van het te laat komen

?? Stappenplan

Stap 1 – Database maken

- Maak een database en een tabel `meldingen` met de volgende velden:
  - `id` (INT, AUTO_INCREMENT, PRIMARY KEY)
  - `student` (VARCHAR)
  - `klas` (VARCHAR)
  - `minuten` (INT)
  - `reden` (TEXT)
- Voeg zelf enkele testregels toe via PHPMyAdmin.

Stap 2 – Read: overzicht maken

- Maak `read.php` waarin je alle meldingen toont in een HTML-tabel.
- Gebruik `require 'connection.php'` voor de databaseverbinding.
- Gebruik `query()` en `fetchAll()` om gegevens op te halen.
- Voeg bovenaan knoppen toe voor “toevoegen” en “zoeken” (optioneel).

Stap 3 – Create: melding toevoegen

- Maak `create.php` met een formulier.
- Voeg invoervelden toe voor student, klas, minuten en reden.
- Sla de gegevens op met een `INSERT`-query via `prepare()` en `execute()`.

- Controleer invoer:
  - geen lege velden
  - minuten is een positief getal
- Toon een foutmelding bij ongeldige invoer en een succesmelding bij correcte invoer.

Stap 4 – Delete: melding verwijderen

- Voeg op `read.php` een knop “verwijder” toe per rij.
- Laat deze verwijzen naar `delete.php?id=...`.
- Toon eerst een bevestigingspagina met de gegevens van de melding.
- Voer pas na bevestiging de `DELETE`-query uit.

Stap 5 – Update

- Voeg op `read.php` een knop “wijzig” toe per rij.
- Laat deze verwijzen naar `update.php?id=...`.
- Haal de bestaande gegevens op met een `SELECT`-query.
- Toon een formulier met ingevulde waarden.
- Werk de gegevens bij met een `UPDATE`-query.

? Reflectie

- Wat heb je geleerd over het werken met databases in PHP?
- Wat ging goed en waar liep je tegenaan?
- Wat zou je de volgende keer anders aanpakken?
- Hoe zorg je voor veilige verwerking van gebruikersinvoer?

? Inleveren

- Screenshot van je overzichtspagina (`read.php`).
- Screenshot van je invoerpagina (`create.php`).
- Screenshot van je wijzigpagina (`update.php`).

- Alle PHP-bestanden, SQL-export en eventuele CSS.
- Een reflectieverslag (.txt of .pdf).
- Wees voorbereid om je code mondeling toe te lichten.

? Puntentelling

Je moet minimaal **81 van de 100 punten** halen om te slagen.

Punten	Onderdeel
10	**Database**: juiste tabel, velden en datatypes.
10	**Read**: overzicht werkt correct.
10	**Layout**: duidelijke en verzorgde opmaak.
20	**Create**: toevoegen werkt inclusief validatie.
10	**Validatie**: ongeldige invoer wordt afgehandeld.
10	**Delete**: verwijderen werkt met bevestiging.
20	**Update**: aanpassen werkt correct.
10	**Codekwaliteit & veiligheid**: gebruik van prepared statements en nette structuur.