Eksamensoppgave i Databasesystemer

Forfattere: Ida Tollaksen (233094) & Emil Berglund (233099)

Gruppe 40 – Canvas

Høgskolen i Østfold – 15.03.2024

HVA VI MANGLER:

Inkluder også potensielle sql-spørringer som må til for å opprette views. Disse spørringene skal forklares og inkluderes resultater av.

* ALLE ChatGPT spørringen og besvarelser må være med
* Forklar og fyll ut om view-et: «Utøver økt oversikt»
* Ev. inkludere flere spørringer til views
* Ev. lage flere views

Innholdsfortegnelse

[Introduksjon: 2](#_Toc161087609)

[Kravspesifikasjon: 3](#_Toc161087610)

[Normalisering: 5](#_Toc161087611)

[ER modellering: 6](#_Toc161087612)

[Dummy data 7](#_Toc161087613)

[Skjermbilder av tabeller 8](#_Toc161087614)

[SQL spørringer 12](#_Toc161087615)

[Avslutning: 18](#_Toc161087616)

[Referanseliste 19](#_Toc161087617)

# Introduksjon:

I denne oppgaven skal vi utforme en database for treningsplanlegging og registrering. Målet er å utvikle et robust system som kan brukes av trenere og utøvere til å strukturere, overvåke og analysere treningsøkter og fremgang over tid.

Kunden ønsker et effektivt system der trenere kan lage individuelle treningsplaner for utøvere, med mulighet for å inkludere ulike typer økter som kondisjonsøkter, styrkeøkter og konkurranser. Det er viktig at trenere kan gjenbruke tidligere øvelser ved opprettelse av nye økter, og at utøvere har tilgang til forklaringer på hvordan hver øvelse skal gjennomføres.

Basert på kundens behov ønsker vi å utvikle en database som gir utøvere mulighet til å se og registrere treningsøkter, inkludert detaljer som distanse, vekt, pulssoner og kommentarer. Denne informasjonen vil være tilgjengelig for både utøvere og trenere, slik at de kan evaluere fremgang over tid.

Vil vi følge en strukturert tilnærming som inkluderer kravspesifikasjon, normalisering av databasen, design av en Entitet-Relasjon (ER) modell, implementering av databasen i MySQL og til slutt fylling av databasen med dummy data.

Videre skal vi utforme SQL-spørringer og views for å demonstrere funksjonaliteten til systemet. Dette kan eksempelvis være progresjon, treningsprogrammer eller treningsøkter. Dette vil gi brukerne et verktøy for å analysere og forbedre treningsregimet og prestasjonene sine over tid.

# Kravspesifikasjon:

Brukere – Brukeradministrasjon

* Databasesystemet skal ha brukere med unik ID i tillegg til relevant informasjon som «fornavn», «etternavn», «alder» og kjønn.
* Brukerne skal differensieres som enten "utøver" eller "trener".

Økt – Overblikk over kommende/tidligere økter

* Tabellen «økt» skal si hvordan type økt en utøver skal gjøre, og da per økt.
* Hver økt får sin egen ID, med «økttype», «dato» i tillegg til «brukerID» som sier noe om hvilke bruker som skal gjøre hvilke økt til hvilken dato.
* En økt kan være en kondisjonsøkt, en styrkeøkt eller en konkurranse.
* Det skal være mulig å skrive kommentarer til hver økt (som en dagbok).

Øvelser – Oversikt over øvelser

* Tabellen «øvelser» skal inneholde alle øvelsene en utøver kan gjøre, eller som en trener kan tildele.
* Hver øvelse får tildelt en unik ID, og har et navn og en beskrivelse.

Øvelselogg – Logg av treningsøkt (treningsøvelser)

* Tabellen «Øvelselogg» skal være logger over alle øvelser en utøver gjør i løpet av en økt.
* Hver logg får tildelt en unik ID, og inkluderer brukerens ID, øvelsens ID, dato og puls-sone.
* Avhengig av typen øvelse, kan det også legges til informasjon om repetisjoner og vekt, eller distanse.

Programmer – Oversikt over opprettede programmer

* Tabellen «Programmer» skal vise til opprettede programmer.
* Hvert program får tildelt en unik ID og et beskrivende navn, som kan referere til ukedager eller annet.
* Tabellen viser også oppvarming og avslutning for hvert program.

Program øvelser – Øvelser tilhørende programmer

* Tabellen "Program\_øvelser" er en oppslagstabell som knytter øvelser til programmer.
* Tabellen skal inneholde en «program\_id» som skal matche «program\_id» i tabellen «programmer». I tillegg skal det også være «ØvelseID» som sier noe om hvilke øvelser som skal gjøres.

Ukeoversikt – Oversikt over hvilken plan utøvere følger

* Tabellen “Ukeoversikt” skal inneholde informasjon om hvilken ukeplan en spesifikk utøver følger, navnet/id-en på treneren som tildelte/opprettet denne planen, i tillegg til start og sluttdato for programmet.
* Hver ukeoversikt skal ha sin egen unike ID.

Ukeplan – Henvisning til program

* Tabellen «ukeplan» skal være en oppslagstabell som kan henvises til for å vise hvilke program som skal brukes.
* Den skal ha en primærnøkkel «PlanID» og en foreign-key «ProgramID» som refererer til programmer.

Ukeplan\_ukedager – Henvisning til hvilke plan som skal brukes til ulike datoer

* Tabellen «ukeplan\_ukedager» skal være en oppslagstabell for å se hvilke plan man skal bruke i henhold til datoen man skal trene.
* Tabellen skal ha en PlanID som referer til «ukeplan»
* Tabellen skal ha en «ukedag» som viser til datoen hvor planene skal brukes.

# Normalisering:

For å normalisere modellen vår til normalform BCNF startet vi med å samle all data i et stort Excel-ark.

1NF – Første Normalform

For å få modellen vår til 1NF startet vi med å sørge for at alle verdiene våre var atomære, altså at vi ikke hadde lister i koloner og at all data i en kollene var av samme datatype. Eksempelvis kunne vi ikke ha fornavn og etternavn i samme kolonne, men måtte separere disse. Deretter undersøkte vi hvilken data som blir redundant og dermed tar opp unødvendig lagringsplass i databasen. Vi evaluerte så hvordan man kan opprette separate tabeller for å lagre informasjonen mer effektivt.

2NF – Andre Normalform

Da vi jobbet med å få database modellen vår til andre normalform, måtte vi fjerne eventuelle partielle avhengigheter. En partiell avhengighet oppstår når en kolonne er avhengig av en del av primærnøkkelen, men ikke hele nøkkelen. Dette fører ofte til et rotete forhold mellom kolonnene innad i tabellen i tillegg til at det skaper en unødvendig rotete struktur.

3NF – Tredje Normalform

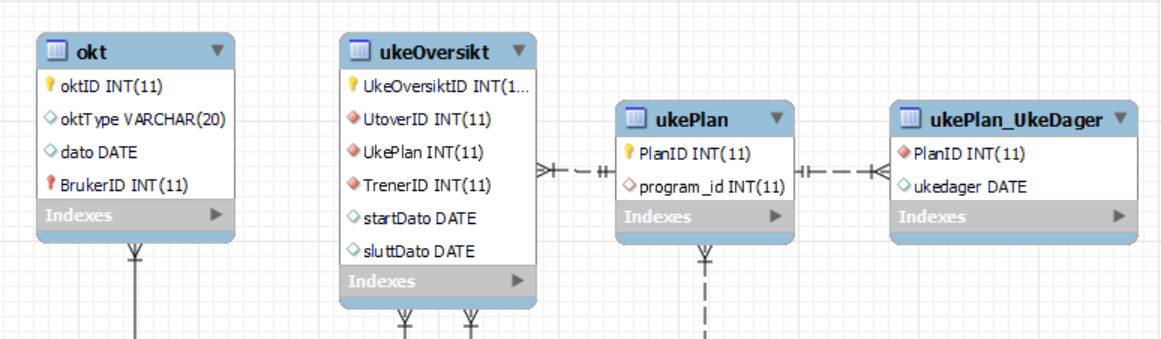
For å få modellen fra andre til tredje normalform måtte vi fjerne alle transitive avhengigheter. Transitive avhengigheter oppstår når to kolonner i en tabell er gjensidig avhengige av hverandre og i tillegg avhenger av primærnøkkelen. Når en tabell inneholder transitive avhengigheter, inneholder den mest sannsynlig også redundante data, noe som tar unødvendig lagringsplass i databasen.

BCNF – Boyce-Codd Normalform

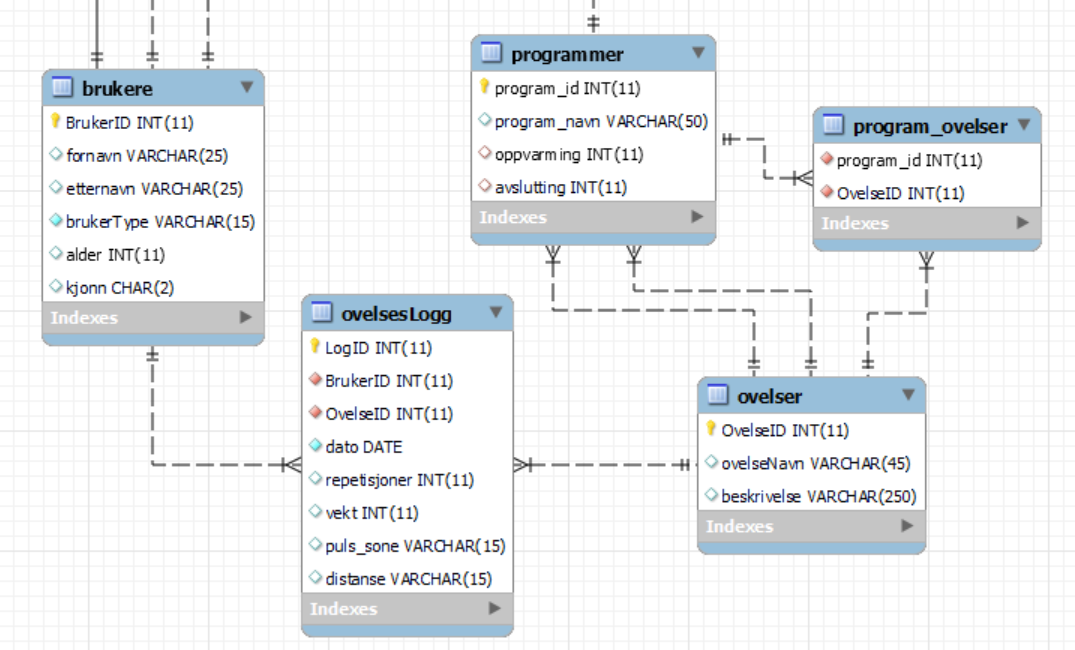
BCNF, eller Boyce-Codd normalform, er en normaliseringsform som bygger på 3NF. Denne normalformen er enda mer restriktiv enn tredje normalform og sikter mot å oppnå en enda bedre database struktur. For å få modellen vår til BCNF måtte vi sørge for at alle minimale determinanter var kandidatnøkler. Dette gjorde vi ved å sikre at alle våre funksjonelle avhengigheter var en triviell funksjonell avhengighet.

Ved å oppnå BCNF på databasen vår kunne vi sikkert si at alle funksjonelle avhengigheter i databasen var godt definerte og at vi ikke hadde noe redundant data. Dette gir oss en veldig god databasestruktur som er lett å vedlikeholde over tid.

# ER modellering:







# Dummy data

Alle våre tabeller er fylt med dummy-data hvor noen er fylt manuelt, og noen er fylt av ChatGPT, med instrukser fra oss. Da ChatGPT ble brukt ble den brukt til å både skrive INSERT INTO scripts, og andre ganger bare informasjonen som vi etterspurte i spørringen. Ett eksempel på dette ligger under, hvor resten av spørringene fra ChatGPT med resultater ligger vedlagt etter litteraturlisten.

Under spurte vi ChatGPT om å lage 500 tilfeldig profiler til «brukere»-tabellen vår med parameterne som vi hadde bestemt på forhånd (brukerID, fornavn, etternavn, brukertype, alder, kjonn), med første linje som eksempel. Det kan argumenteres for at 500 kanskje var litt mye, og dermed unødvendig, men vi ville være på den sikre siden, og tok derfor med alle 500.

På grunnlaget av at det ble 500 brukere, fikk vi ChatGPT til å lage INSERT spørringene for oss. Ettersom det også ble såpas mange linjer, har vi valgt for oversiktens skyld å bare inkludere de 9 første. Resterende brukere har en økende og unik ID, samt ulike/tilfeldige fornavn, etternavn, brukertyper, alder og kjønn.

**INSERT** **INTO** brukere (brukerID, fornavn, etternavn, brukerType, alder, kjonn) **VALUES**  
(1, 'Nancy', 'Gerlach', 'Kunde', 26, 'D'),  
(2, 'Marcella', 'Parisian', 'Trener', 26, 'D'),  
(3, 'Guillermo', 'Homenick', 'Kunde', 38, 'D'),  
(4, 'Jared', 'West', 'Kunde', 29, 'M'),  
(5, 'Janie', 'Hilpert', 'Kunde', 60, 'D'),  
(6, 'Francis', 'Champlin', 'Trener', 24, 'M'),  
(7, 'Ramon', 'Osinski', 'Trener', 27, 'M'),  
(8, 'Stewart', 'Howell', 'Kunde', 21, 'M'),  
(9, 'Sue', 'Gutkowski', 'Trener', 41, 'D'),

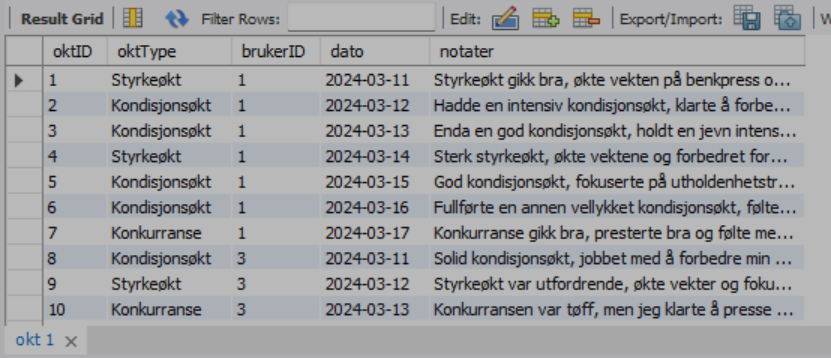
# Skjermbilder av tabeller

Brukere:

Et bilde som inneholder tekst, skjermbilde, nummer, Font

Automatisk generert beskrivelse

Økt:

Øvelser:

Et bilde som inneholder tekst, Font, nummer, skjermbilde

Automatisk generert beskrivelse

Øvelseslogg:

Et bilde som inneholder tekst, skjermbilde, nummer, programvare

Automatisk generert beskrivelse

Program øvelser:

Et bilde som inneholder tekst, skjermbilde, nummer, Font

Automatisk generert beskrivelse

Programmer:

Et bilde som inneholder tekst, skjermbilde, Font, nummer

Automatisk generert beskrivelse

Ukeoversikt:

Et bilde som inneholder tekst, skjermbilde, nummer, Font

Automatisk generert beskrivelse

Ukeplan:

Et bilde som inneholder tekst, skjermbilde, nummer, Font

Automatisk generert beskrivelse

Ukeplan-ukedager:

Et bilde som inneholder tekst, skjermbilde, Font, nummer

Automatisk generert beskrivelse

# SQL spørringer

Views:

Bruker\_oversikt\_uke:

**CREATE** VIEW bruker\_oversikt\_uke **AS**

**SELECT**

ukeOversikt.UtoverID **AS** BrukerID,

**CONCAT**(brukere.fornavn, ' ', brukere.etternavn) **AS** UtøverNavn,

ukeOversikt.TrenerID **AS** TrenerID,

**CONCAT**(trener.fornavn, ' ', trener.etternavn) **AS** TrenerNavn,

ukePlan\_UkeDager.ukedager **AS** Ukedag,

ukeOversikt.UkePlan **AS** PlanID,

programmer.program\_navn **AS** ProgramNavn,

okt.oktID,

okt.oktType

**FROM**

ukeOversikt

**JOIN**

brukere **ON** ukeOversikt.UtoverID = brukere.BrukerID

**JOIN**

brukere **AS** trener **ON** ukeOversikt.TrenerID = trener.BrukerID

**JOIN**

ukePlan **ON** ukeOversikt.UkePlan = ukePlan.PlanID

**JOIN**

ukePlan\_UkeDager **ON** ukePlan.PlanID = ukePlan\_UkeDager.PlanID

**JOIN**

programmer **ON** ukePlan.program\_id = programmer.program\_id

**JOIN**

okt **ON** ukePlan\_UkeDager.ukedager = okt.dato;

Et bilde som inneholder tekst, skjermbilde, nummer, Font

Automatisk generert beskrivelse

Bruker\_oversikt\_uke er et view laget for å illustrere en oversiktsplan for alle brukere. Oversikten inkluderer alle brukere som har fått tildelt en plan. Man kan derfor se hvilke bruker som har hvilke trener, hvilke plan den personen skal følge til hvilken dag, og også få vite hvordan økt det er. View-et er laget for å være allsidig. En eksempel-spørringen mot view-et kan for eksempel være å filtrere på BrukerID. Da vil man få opp relevant informasjon til utøveren i fokus.

I SELECT-setningen velges forskjellige kolonner og fornavn og etternavn slåes sammen ved hjelp av CONCAT-funksjonen for å kombinere fornavn og etternavn til både utøveren og treneren.

Det utføres en rekke JOIN-operasjoner for å koble tabellene sammen basert på deres relasjoner. For eksempel, brukes JOIN til å kombinere ukeOversikt-tabellen med brukere-tabellen basert på BrukerID. JOIN brukes også til å kombinere ukePlan\_UkeDager-tabellen med okt-tabellen basert på datoen for økten.

Bruker\_progresjon:

**CREATE** VIEW bruker\_progresjon **AS**

**SELECT**

ovelsesLogg.BrukerID,

**CONCAT**(brukere.fornavn, ' ', brukere.etternavn) **AS** UtøverNavn,

ovelsesLogg.OvelseID,

ovelser.ovelseNavn,

**MIN**(ovelsesLogg.dato) **AS** forsteDato,

**MAX**(ovelsesLogg.dato) **AS** sisteDato,

**MIN**(ovelsesLogg.vekt) **AS** minVekt,

**MAX**(ovelsesLogg.vekt) **AS** maxVekt,

**MIN**(ovelsesLogg.repetisjoner) **AS** minRepetisjoner,

**MAX**(ovelsesLogg.repetisjoner) **AS** maxRepetisjoner

**FROM**

ovelsesLogg

**JOIN**

brukere **ON** ovelsesLogg.BrukerID = brukere.BrukerID

**JOIN**

ovelser **ON** ovelsesLogg.OvelseID = ovelser.OvelseID

**WHERE**

ovelsesLogg.OvelseID **NOT** **IN** (**1**, **3**)

**GROUP** **BY**

ovelsesLogg.BrukerID, ovelsesLogg.OvelseID;

Et bilde som inneholder tekst, skjermbilde, nummer, Font

Automatisk generert beskrivelse

Bruker\_progresjon er et view som gir en detaljert oversikt over brukerens fremgang i treningsprogrammet. Visningen er utformet for å vise informasjon om hver brukers fremdrift i deres treningsplan og dermed de ulike øvelsene. Man har full oversikt over alle brukere, hvilke øvelse det gjelder, første- og sistedato og selvfølgelig vekten og repetisjonene med minimums og maksimumsverdiene en utøver har klart.

I SELECT-setningen er ulike kolonner valgt for å gi en omfattende oversikt over brukerens progresjon. Fornavn og etternavn kombineres ved hjelp av CONCAT-funksjonen for å gi fullstendige navn for utøveren.

Det utføres JOIN-operasjoner mellom disse tabellene basert på deres relasjoner. JOIN brukes blant annet til å kombinere ukeOversikt-tabellen med brukere-tabellen basert på BrukerID, og JOIN brukes til å koble ukePlan\_UkeDager-tabellen med okt-tabellen basert på øktens dato.

Dette view-et er utformet for å være allsidig, slik at man kan filtrere på ulike kriterier for å få en mer spesifikk visning av brukerens progresjon. For eksempel kan man filtrere på BrukerID og OvelseID for å få relevant informasjon om en bestemt utøver og eventuell øvelse og dermed deres fremgang i treningsprogrammet.

Utøver økt oversikt:

**CREATE** VIEW Utover\_Okt\_Overview **AS**

**SELECT**

o.oktID,

o.oktType,

o.dato **AS** Dato,

o.brukerID **AS** BrukerID,

u.fornavn **AS** UtøverFornavn,

u.etternavn **AS** UtøverEtternavn,

ou.ovelseNavn **AS** Øvelse,

ou.beskrivelse **AS** ØvelsesBeskrivelse

**FROM**

okt o

**JOIN**

brukere u **ON** o.brukerID = u.BrukerID

**JOIN**

ovelsesLogg ol **ON** o.dato = ol.dato

**JOIN**

ovelser ou **ON** ol.OvelseID = ou.OvelseID;

Et bilde som inneholder tekst, skjermbilde, nummer, Font

Automatisk generert beskrivelse

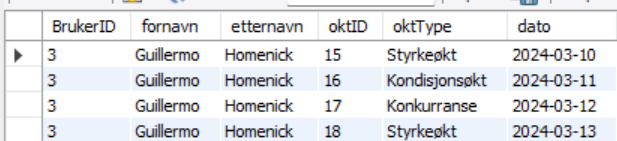
Spørringer

Spørring:

**SELECT** b.BrukerID, b.fornavn, b.etternavn, o.oktID, o.oktType, o.dato  
**FROM** brukere **AS** b  
**JOIN** okt **AS** o **ON** b.BrukerID = o.BrukerID  
**WHERE** o.BrukerID = '3';

Resultat:

Henter informasjon fra brukere og økt tabellen om brukeren som har bruker id 3.

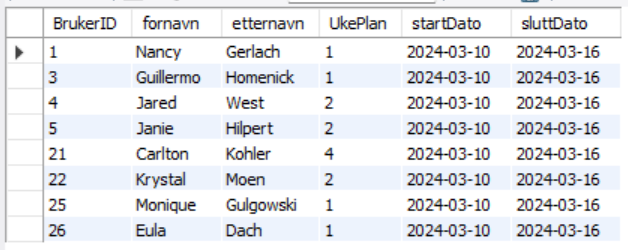


Spørring:

**SELECT** b.BrukerID, b.fornavn, b.etternavn, u.UkePlan, startDato, sluttDato  
**FROM** brukere **AS** b  
**JOIN** ukeOversikt **AS** u **ON** b.BrukerID = u.UtoverID  
**WHERE** startDato = '2024-03-10' **and** sluttDato = '2024-03-16';

Resultat:

Henter ut informasjon om brukere som utfører en ukeplan som har start dato “2024-03-10” og slutt dato “2024-03-16”.



Spørring:

**SELECT**

(**SELECT** **AVG**(alder) **FROM** brukere **WHERE** brukerType = 'Trener') **AS** gjennomsnittsalderUtover,

(**SELECT** **AVG**(alder) **FROM** brukere **WHERE** brukerType = 'Utover') **AS** gjennomsnittalderTrener;

Resultat:

Henter ut gjennomsnittsalder for både trenere og utøvere i to separate kolonner.

Et bilde som inneholder tekst, Font, line, hvit

Automatisk generert beskrivelse

Spørring:

**SELECT** OvelseID, ovelseNavn

**FROM** ovelser

**WHERE** OvelseID **NOT** **IN** (**SELECT** **DISTINCT** OvelseID **FROM** ovelsesLogg);

Resultat:

Henter ut alle øvelser med ID som ikke har blitt skrevet logg på.

Et bilde som inneholder tekst, skjermbilde, Font, nummer

Automatisk generert beskrivelse

Spørring:

**SELECT** \* **FROM** Utover\_Okt\_Overview **WHERE** BrukerID = **1**;

Resultat:

Viser alle øktene til Utøveren med ID = 1. Her hentes informasjonen fra view-et som er vist over (Utover\_Okt\_Overview).

Et bilde som inneholder tekst, Font, nummer, skjermbilde

Automatisk generert beskrivelse

# Avslutning:

Gjennom denne oppgaven har vi laget et databasesystem til trenere og utøvere ved å designe og implementere en strukturert treningsplanlegging og registreringsdatabase. Dette oppnådde vi ved å utvikle et system som gir trenere og utøvere muligheten til å enkelt opprette, redigere og holde oversikt over treningsøkter og fremgang over tid.

Oppgaven vår inkluderer en grundig kravspesifikasjon, normalisering av databasen, design av en Entitet-Relasjon (ER) modell, implementering i MySQL og fylling av databasen med dummy data. Videre har vi utformet SQL-spørringer og views for å demonstrere systemets funksjonalitet, noe som illustrerer hvordan brukerne kan bruke databasen for å forbedre sitt treningsregime og prestasjoner.

Gjennom denne oppgaven har vi derfor oppnådd at vårt system møter de spesifikke behovene til både trenere og utøverne. I tillegg gir programmet vårt dem et pålitelig verktøy for å optimalisere sin treningsstruktur og oppnå sine mål.

# Referanseliste

OpenAI. (n.d.). *ChatGPT*. Chat.openai.com; OpenAI. Hentet 8. mars 2024, fra <https://chat.openai.com/>

Kojevnikov, A. (n.d.). *Source code beautifier / syntax highlighter – convert code snippets to HTML «hilite.me*. Hilite.me. Hentet 10. mars 2024, fra <http://hilite.me/>

ChatGPT spørring 1:

*Basert på dette, generer opp mer tilfeldig dummy data til tabellen, verdiene under kjon skal enten være “D” eller “M”: INSERT INTO bruker (brukerID, fornavn, etternavn, brukerType, alder, kjonn) VALUES (1, 'Nancy', 'Gerlach', 'Kunde', 26, 'D')*

Svar:

Here's an SQL INSERT INTO script for the "bruker" table:

**INSERT** **INTO** bruker (brukerID, fornavn, etternavn, brukerType, alder, kjonn) **VALUES**  
(1, 'Nancy', 'Gerlach', 'Kunde', 26, 'D'),  
(2, 'Marcella', 'Parisian', 'Trener', 26, 'D'),  
(3, 'Guillermo', 'Homenick', 'Kunde', 38, 'D'),  
(4, 'Jared', 'West', 'Kunde', 29, 'M'),  
(5, 'Janie', 'Hilpert', 'Kunde', 60, 'D'),  
(6, 'Francis', 'Champlin', 'Trener', 24, 'M'),  
(7, 'Ramon', 'Osinski', 'Trener', 27, 'M'),  
(8, 'Stewart', 'Howell', 'Kunde', 21, 'M'),  
(9, 'Sue', 'Gutkowski', 'Trener', 41, 'D'),  
(10, 'Oscar', 'Kub', 'Trener', 37, 'M'),  
(11, 'Frankie', 'Kilback', 'Trener', 56, 'D'),  
(12, 'Jared', 'Walter', 'Trener', 34, 'D')

This script will insert all the provided data into the "bruker" table, assuming that the columns are correctly set up in your database.

ChatGPT spørring 2:

*Kan du legge til dummydata i disse kommentarene maks 200 tegn. Egentlig bare en kort oppsummering på hvordan en hypotetisk treningsøkt gikk (etter dato) : 1 Styrkeøkt 1 2024-03-11 2 Kondisjonsøkt 1 2024-03-12 3 Kondisjonsøkt 1 2024-03-13 4 Styrkeøkt 1 2024-03-14 5 Kondisjonsøkt 1 2024-03-15 6 Kondisjonsøkt 1 2024-03-16 7 Konkurranse 1 2024-03-17 8 Kondisjonsøkt 2 2024-03-11 9 Styrkeøkt 2 2024-03-12 10 Konkurranse 2 2024-03-13 11 Kondisjonsøkt 2 2024-03-14 12 Styrkeøkt 2 2024-03-15 13 Konkurranse 2 2024-03-16 14 Styrkeøkt 2 2024-03-17 15 Styrkeøkt 3 2024-03-11 16 Kondisjonsøkt 3 2024-03-12 17 Konkurranse 3 2024-03-13 18 Styrkeøkt 3 2024-03-14 19 Kondisjonsøkt 3 2024-03-15 20 Konkurranse 3 2024-03-16 21 Konkurranse 3 2024-03-17*

Svar:

-- 1 Styrkeøkt gikk bra, økte vekten på benkpress og følte meg sterk.

-- 2 Hadde en intensiv kondisjonsøkt, klarte å forbedre tidene mine.

-- 3 Enda en god kondisjonsøkt, holdt en jevn intensitet gjennom hele økten.

-- 4 Sterk styrkeøkt, økte vektene og forbedret formen min.

-- 5 God kondisjonsøkt, fokuserte på utholdenhetstrening.

-- 6 Fullførte en annen vellykket kondisjonsøkt, følte meg energisk.

-- 7 Konkurranse gikk bra, presterte bra og følte meg tilfreds.

-- 8 Solid kondisjonsøkt, jobbet med å forbedre min utholdenhet.

-- 9 Styrkeøkt var utfordrende, økte vekter og fokuserte på form.

-- 10 Konkurransen var tøff, men jeg klarte å presse meg gjennom.

-- 11 Kondisjonsøkten var intens, klarte å slå mine personlige rekorder.

-- 12 Sterk styrkeøkt, forbedret styrken og teknikken min.

-- 13 Konkurransen var krevende, men jeg ga mitt beste.

-- 14 Styrkeøkten var utfordrende, men jeg følte meg sterk og fokusert.

-- 15 Gjennomførte en god styrkeøkt, økte vektene gradvis.

-- 16 Kondisjonsøkten var bra, klarte å opprettholde en god intensitet.

-- 17 Konkurransen gikk bra, presterte godt under press.

-- 18 Styrkeøkten var intens, fokuserte på kjerneøvelser.

-- 19 Jobbet med utholdenheten i dagens kondisjonsøkt.

-- 20 Konkurransen var tøff, men jeg kjempet meg gjennom.

-- 21 Konkurranse gikk bra, følte meg godt forberedt og fokusert.

ChatGPT spørring 3:

*Basert på dette, generer opp mer tilfeldig dummy data til tabellen, hvis det er verdi i distansekolonene skal det ikke være verdier i repitisjoner og vekt kollonen og visa versa: INSERT INTO bruker (LogID, BrukerID, OvelseID, dato, repitisjoner, vekt, puls\_sone, distanse) VALUES (1, 1, 1, ‘2024-03-11’, ‘NULL’, ‘NULL’, ‘120-140’, ‘2km’ ), (2, 1, 9, ‘2024-03-11', 8, 20, ‘100-120’, ‘NULL’)*

Svar:

Here's an example of generating random dummy data for your table based on the criteria you provided:

**INSERT** **INTO** bruker (LogID, BrukerID, OvelseID, dato, repitisjoner, vekt, puls\_sone, distanse)  
**VALUES**  
(1, 1, 1, '2024-03-11', NULL, NULL, '120-140', '2km'),  
(2, 1, 9, '2024-03-11', 8, 20, '100-120', NULL),  
(3, 1, 9, '2024-03-11', 8, 30, '100-120', NULL),  
(4, 1, 9, '2024-03-11', 10, 35, '100-120', NULL),  
(5, 1, 10, '2024-03-11', 8, 20, '100-120', NULL),  
(6, 1, 10, '2024-03-11', 10, 20, '100-120', NULL),  
(7, 1, 10, '2024-03-11', 10, 20, '100-120', NULL),  
(8, 1, 11, '2024-03-11', 8, 20, '100-120', NULL);

This script inserts dummy data into the bruker table based on the provided data, following the pattern where either repetisjoner and vekt columns have values or the distanse column has a value for each row. Adjust as needed for your specific requirements.