Ukesoppgave 03 - MySql

Denne oppgaven gir studentene en omfattende innføring i MySQL-databaseadministrasjon og integrasjon med en Node.js-applikasjon. Studentene starter med å installere MySQL Server og MySQL Workbench på både Windows og macOS, med et alternativ for macOS-brukere til å bruke Homebrew. Deretter lærer de å konfigurere MySQL Server, opprette en applikasjonsbruker, og sette opp en database med tilhørende tabeller, samtidig som de får en grunnleggende forståelse av tabellkomponenter. Videre får studentene praktisk erfaring med å utføre CRUD-operasjoner (Create, Read, Update, Delete) på databasen ved hjelp av SQL-spørringer. I neste del øver de på å integrere MySQL med en Node.js Express-applikasjon, hvor de setter opp API-endepunkter for CRUD-operasjoner. Til slutt introduseres de for Prisma ORM, et moderne ORM-verktøy, som tilbyr en type-sikker og forenklet tilnærming til databaseinteraksjon i applikasjoner. Gjennom disse delene får studentene både praktisk erfaring og teoretisk forståelse av databaseadministrasjon og backend-utvikling.

Del 1: Installere MySQL Server og MySQL Workbench

Før vi begynner, er det viktig å forstå hva vi installerer:

- **MySQL Server:** Dette er selve databaseserveren som lagrer, organiserer og administrerer dataene dine. Den håndterer alle databaseforespørsler og sikrer at dataene er tilgjengelige og konsistente.

- **MySQL Workbench:** Dette er et grafisk brukergrensesnitt (GUI) verktøy som gjør det enklere å administrere MySQL-serveren. Med Workbench kan du designe databaser, skrive og kjøre SQL-spørringer, samt administrere brukere og rettigheter uten å bruke kommandolinjen.

Installere på Windows

Følg disse trinnene for å installere MySQL Server og MySQL Workbench på en Windows-maskin:

Trinn 1: Last ned MySQL Installer

1. **Gå til MySQLs offisielle nedlastingsside:**
   - Åpne din nettleser og naviger til MySQL Community Downloads.
2. **Velg riktig installer:**
   - Du vil se to alternativer:
     - **MySQL Installer (webversjon):** Dette er en liten nedlastningsfil som laster ned de nødvendige komponentene under installasjonen.
     - **MySQL Installer (fullversjon):** Dette er en større fil som inneholder alle nødvendige komponenter.
   - For de fleste brukere anbefales webversjonen. Klikk på "Download" ved siden av "MySQL Installer Web".
3. **Logg inn eller fortsett uten å logge inn:**
   - Du blir bedt om å logge inn eller registrere deg, men du kan klikke på "No thanks, just start my download." for å fortsette uten å logge inn.

Trinn 2: Kjør MySQL Installer

1. **Åpne den nedlastede filen:**
   - Finn den nedlastede `.msi`-filen i nedlastingsmappen din og dobbeltklikk for å starte installasjonsprosessen.
2. **Velg installasjonstype:**
   - Du vil bli presentert med flere installasjonsalternativer:
     - **Developer Default:** Anbefalt for utviklere, inkluderer alle nødvendige verktøy.
     - **Server only:** Kun MySQL Server installeres.
     - **Client only:** Kun klientverktøy som Workbench installeres.
     - **Full:** Alle MySQL-produkter installeres.
     - **Custom:** Tillater deg å velge spesifikke komponenter.
   - **Anbefaling:** Velg **Developer Default** for å sikre at både server og nødvendige verktøy installeres.
3. **La installasjonsveiviseren utføre nødvendige forberedelser:**
   - Installereren vil sjekke om du mangler noen nødvendige avhengigheter. Følg eventuelle instruksjoner for å installere manglende komponenter.
4. **Installer produktene:**
   - Klikk på "Execute" for å begynne installasjonen av de valgte komponentene. Dette kan ta noen minutter.

Trinn 3: Konfigurer MySQL Server

1. **Start konfigurasjonsveiviseren:**
   - Etter installasjonen åpnes MySQL Configuration Wizard automatisk. Hvis ikke, kan du åpne den fra Start-menyen under MySQL > MySQL Installer > Configure.
2. **Velg konfigurasjonstype:**
   - Velg "Standalone MySQL Server / Classic MySQL Replication" og klikk "Next".
3. **Angi produktoppdateringer:**

- Velg om du vil aktivere produktoppdateringer via MySQL eller ikke. For nybegynnere anbefales det å la standardinnstillingene stå og klikk "Next".

4. **Sett opp MySQL Server:**

   - **Velg versjon:** Velg den nyeste stabile versjonen og klikk "Next".
   - **Konfigurer server:** La de fleste innstillingene være standard. Klikk "Next".

5. **Angi root-passord:**

   - **Root-konto:** Dette er administratorkontoen for MySQL. Velg et sterkt passord og bekreft det.
   - **Opprett brukere:** Du kan legge til ekstra brukere her, men for nå er det tilstrekkelig å sette root-passordet.
   - Klikk "Next".

6. **Fullfør konfigurasjonen:**

   - Klikk "Execute" for å anvende konfigurasjonene.
   - Når prosessen er fullført, klikk "Finish".

Trinn 4: Start og Test MySQL Server

1. **Start MySQL Workbench:**

   - Etter installasjonen kan du åpne MySQL Workbench fra Start-menyen under MySQL > MySQL Workbench.

2. **Opprett en ny tilkobling:**

   - Klikk på "+"-ikonet ved siden av "MySQL Connections" for å opprette en ny tilkobling.
   - **Connection Name:** Gi tilkoblingen et navn, f.eks. "Local MySQL".
   - **Hostname:** `localhost` (standardinnstilling).
   - **Port:** `3306` (standard MySQL-port).
   - **Username:** `root`.
   - Klikk "Test Connection" og skriv inn root-passordet du opprettet under installasjonen.
   - Hvis tilkoblingen er vellykket, klikker du "OK" for å lagre tilkoblingen.

3. **Bekreft at serveren kjører:**

   - Dobbeltklikk på den nylig opprettede tilkoblingen for å koble til MySQL-serveren. Hvis du ser en vellykket tilkobling, er installasjonen fullført.

Installere på macOS

Følg disse trinnene for å installere MySQL Server og MySQL Workbench på en macOS-maskin. Vi inkluderer både en metode ved bruk av Homebrew og en tradisjonell nedlasting.

Metode 1: Bruke Homebrew (Anbefalt)

Homebrew er en populær pakkebehandler for macOS som gjør installasjon og administrasjon av programvare enklere.

Trinn 1: Installer Homebrew (hvis ikke allerede installert)

1. **Åpne Terminal:**

   - Finn Terminal under Programmer > Verktøy, eller søk etter "Terminal" via Spotlight.

2. **Installer Homebrew:**

   - Kopier og lim inn følgende kommando i Terminal, og trykk Enter:

```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

   - Følg instruksjonene på skjermen for å fullføre installasjonen. Du kan bli bedt om å oppgi ditt macOS-passord.

Trinn 2: Installer MySQL Server via Homebrew

1. **Oppdater Homebrew:**

   - Sørg for at Homebrew er oppdatert ved å kjøre:

```
brew update
```

2. **Installer MySQL:**

   - Kjør følgende kommando for å installere MySQL Server:

```
brew install mysql
```

3. **Start MySQL-tjenesten:**

   - For å starte MySQL-serveren automatisk ved oppstart, kjør:

```
brew services start mysql
```

   - Alternativt kan du starte MySQL manuelt med:

```
mysql.server start
```

4. **Sikre MySQL-installasjonen:**

   - Kjør sikkerhetsinstallasjonen for å sette root-passord og sikre serveren:

```
mysql_secure_installation
```

   - Følg instruksjonene på skjermen. Dette inkluderer å sette et root-passord, fjerne anonyme brukere, og andre sikkerhetsrelaterte innstillinger.

Trinn 3: Installer MySQL Workbench via Homebrew

1. **Installer MySQL Workbench:**

   - Kjør følgende kommando:

```
brew install --cask mysqlworkbench
```

2. **Start MySQL Workbench:**

   - Åpne MySQL Workbench fra Programmer-mappen eller ved å søke etter den via Spotlight.

Trinn 4: Konfigurer MySQL Workbench

1. **Opprett en ny tilkobling:**
   - Klikk på "+"-ikonet ved siden av "MySQL Connections" for å opprette en ny tilkobling.
   - **Connection Name:** For eksempel, "Local MySQL".
   - **Hostname:** `localhost`.
   - **Port:** `3306`.
   - **Username:** `root`.
   - Klikk "Test Connection" og skriv inn root-passordet du satte under `mysql_secure_installation`.
   - Hvis tilkoblingen er vellykket, klikker du "OK" for å lagre tilkoblingen.

2. **Bekreft at serveren kjører:**
   - Dobbeltklikk på den nylig opprettede tilkoblingen for å koble til MySQL-serveren. Hvis du ser en vellykket tilkobling, er installasjonen fullført.

Metode 2: Tradisjonell Nedlasting

Hvis du foretrekker å laste ned og installere MySQL manuelt, kan du følge disse trinnene:

Trinn 1: Last ned MySQL DMG-fil

1. **Gå til MySQLs offisielle nedlastingsside:**
   - Åpne din nettleser og naviger til MySQL Community Downloads.

2. **Velg MySQL Community Server:**
   - Klikk på "Download" ved siden av "MySQL Community Server".

3. **Velg din macOS-versjon:**
   - Velg riktig macOS-versjon fra listen og klikk "Download".

Trinn 2: Installer MySQL Server

1. **Åpne DMG-filen:**
   - Når nedlastingen er fullført, åpne DMG-filen.

2. **Kjør installasjonsprogrammet:**
   - Dobbeltklikk på installasjonsprogrammet og følg instruksjonene på skjermen.

3. **Sett root-passord:**
   - Under installasjonen vil du bli bedt om å sette et root-passord. Husk dette passordet for senere bruk.

4. **Start MySQL-tjenesten:**
   - Når installasjonen er fullført, start MySQL-tjenesten via System Preferences > MySQL.

Trinn 3: Last ned og Installer MySQL Workbench

1. **Gå til MySQL Workbench nedlastingsside:**

- Naviger til MySQL Workbench Download.

2. **Velg riktig versjon:**
   - Last ned versjonen som passer for din macOS.
3. **Installer MySQL Workbench:**
   - Åpne DMG-filen og dra MySQL Workbench til Programmer-mappen.

Trinn 4: Konfigurer MySQL Workbench

1. **Åpne MySQL Workbench:**
   - Start MySQL Workbench fra Programmer-mappen.
2. **Opprett en ny tilkobling:**
   - Følg de samme trinnene som i Metode 1 for å opprette og teste en ny tilkobling til MySQL-serveren.

Del 2: Konfigurere MySQL Server og Opprette en Database og Tabell

Nå som MySQL Server og MySQL Workbench er installert, er det på tide å konfigurere serveren med en bruker og opprette en database. Deretter skal vi opprette en tabell i denne databasen. En tabell er en strukturert samling av data i rader og kolonner, og hver kolonne representerer en type data, mens hver rad representerer en post.

Trinn 1: Konfigurer MySQL Server med en Applikasjonsbruker

Hvorfor Opprette en Applikasjonsbruker?

Det er god praksis å ikke bruke `root`-kontoen til applikasjoner som trenger tilgang til databasen. I stedet oppretter vi en dedikert bruker med begrensede rettigheter for sikkerhet og kontroll.

Trinn 1.1: Logg Inn på MySQL Workbench

1. **Åpne MySQL Workbench:**
   - Start MySQL Workbench fra Start-menyen (Windows) eller Programmer-mappen (macOS).
2. **Koble til MySQL Server:**
   - Dobbeltklikk på tilkoblingen du opprettet i Del 1 (f.eks. "Local MySQL").
   - Skriv inn `root`-passordet ditt hvis det blir bedt om det.

Trinn 1.2: Opprett en Ny Bruker

1. **Åpne Bruker- og Rettighetsadministrasjon:**
   - I MySQL Workbench, naviger til `Server > Users and Privileges` fra menylinjen.
2. **Legg til en Ny Bruker:**

- Klikk på "Add Account" i nedre venstre hjørne.
- Under "Login Name", skriv inn et brukernavn (f.eks. `app_user`).
- Under "Limit to Hosts Matching", skriv `localhost` for å begrense tilgangen til kun lokal maskin.
- Under "Authentication", skriv inn et sterkt passord og bekreft det.

3. **Tildel Rettigheter:**

   - Gå til fanen "Administrative Roles" og velg "DBA" for full tilgang.
   - Alternativt kan du tildele spesifikke rettigheter under fanene "Schema Privileges" eller "Administrative Roles", avhengig av behov.

4. **Lagre Brukeren:**

   - Klikk "Apply" for å lagre den nye brukeren.

Trinn 1.3: Test den Nye Brukeren

1. **Koble fra den Nåværende Tilkoblingen:**

   - Gå til `File > Close All` for å lukke alle åpne tilkoblinger.

2. **Opprett en Ny Tilkobling med den Nye Brukeren:**

   - Klikk på "+"-ikonet ved siden av "MySQL Connections" for å opprette en ny tilkobling.
   - Gi tilkoblingen et navn (f.eks. "App User Connection").
   - Under "Username", skriv inn brukernavnet du opprettet (f.eks. `app_user`).
   - Klikk "Test Connection" og skriv inn passordet for å teste tilkoblingen.
   - Hvis testen er vellykket, klikk "OK" for å lagre tilkoblingen.

Trinn 2: Opprett en Database

Hva er en Database?

En database er en strukturert samling av data som kan administreres, søkes og hentes av MySQL-serveren. Databasen inneholder en eller flere tabeller, som hver lagrer data om et spesifikt emne.

Trinn 2.1: Opprett en Ny Database

1. **Koble til MySQL Server:**

   - Bruk den nye tilkoblingen du opprettet for `app_user`.

2. **Opprett en Ny Database:**

   - I SQL Editor-vinduet, skriv inn følgende SQL-kommando for å opprette en ny database:

   ```
   CREATE DATABASE student_database;
   ```

   - Klikk på den gule lyn-ikonet (Execute) eller trykk `Ctrl + Enter` for å kjøre kommandoen.

3. **Verifiser Databasen:**

   - Gå til "Schemas" panelet til venstre i Workbench, høyreklikk og velg "Refresh All" for å oppdatere listen. Du skal nå se `student_database` i listen over databaser.

Trinn 3: Opprett en Tabell i Databasen

Hva er en Tabell?

En tabell i en database er en struktur som består av kolonner (felter) og rader (poster). Hver kolonne har et navn og en datatype som spesifiserer hva slags data kolonnen kan inneholde (f.eks. tekst, tall, datoer). Hver rad i tabellen representerer en individuell post.

Trinn 3.1: Velg Databasen

1. **Velg `student_database`:**
   - Sørg for at du er koblet til MySQL-serveren med `app_user`.
   - Skriv følgende kommando for å bruke den nye databasen:

```
USE student_database;
```

Trinn 3.2: Opprett en Ny Tabell

1. **Opprett Tabellen:**
   - Skriv inn følgende SQL-kommando for å opprette en tabell kalt `students`:

```
CREATE TABLE students (
    student_id INT AUTO_INCREMENT PRIMARY KEY,
    first_name VARCHAR(50),
    last_name VARCHAR(50),
    enrollment_date DATE
);
```

   - **Forklaring av Kolonnene:**
     - `student_id`: En unik identifikator for hver student. `INT` betyr at dette er et heltall. `AUTO_INCREMENT` gjør at MySQL automatisk øker verdien for hver ny rad. `PRIMARY KEY` gjør dette feltet til tabellens unike nøkkel.
     - `first_name`: Studentens fornavn. `VARCHAR(50)` betyr at dette er en tekstkolonne som kan inneholde opptil 50 tegn.
     - `last_name`: Studentens etternavn, også en tekstkolonne med opptil 50 tegn.
     - `enrollment_date`: Datoen studenten ble innmeldt. `DATE` betyr at dette feltet lagrer en dato.

2. **Kjør Kommandoen:**
   - Klikk på Execute-ikonet eller trykk `Ctrl + Enter` for å opprette tabellen.

3. **Verifiser Tabellen:**
   - Gå til "Schemas" panelet, under `student_database`, og se etter `students` tabellen under "Tables".

Del 3: Legge til, Oppdatere, Lese og Slette Data fra Tabellen

I denne delen skal vi jobbe med dataene i tabellen du opprettet i Del 2. Vi skal lære hvordan du kan legge til nye rader (poster) i tabellen, oppdatere eksisterende data, lese (hente) data, og slette uønskede rader.

Trinn 1: Legge til Data (INSERT)

Hva er en INSERT-kommando?

INSERT-kommandoen brukes for å legge til nye rader i en tabell. Hver ny rad vil inneholde data som samsvarer med kolonnene i tabellen.

Trinn 1.1: Legg til en Ny Rad i Tabellen

1. **Velg Databasen:**
   - Før du kan legge til data, må du forsikre deg om at du bruker riktig database. Skriv inn:

```
USE student_database;
```

2. **Legg til en Ny Student:**
   - Bruk følgende SQL-kommando for å legge til en ny student i `students`-tabellen:

```
INSERT INTO students (first_name, last_name, enrollment_date)
VALUES ('John', 'Doe', '2024-09-01');
```

   - **Forklaring:**
     - `first_name`: Fornavnet til studenten, her `'John'`.
     - `last_name`: Etternavnet til studenten, her `'Doe'`.
     - `enrollment_date`: Datoen studenten ble innmeldt, her `'2024-09-01'`.
   - **Merk:** Du trenger ikke å inkludere `student_id` fordi det er satt til å AUTO_INCREMENT, og MySQL vil automatisk generere denne verdien.

3. **Kjør Kommandoen:**
   - Klikk på Execute-ikonet eller trykk `Ctrl + Enter` for å kjøre INSERT-kommandoen.

4. **Verifiser Dataene:**
   - Du kan sjekke at dataene ble lagt til ved å lese fra tabellen (se Trinn 3.1 nedenfor).

Trinn 1.2: Legg til Flere Rader

1. **Legg til Flere Studenter:**
   - Du kan legge til flere rader ved å bruke flere INSERT-kommandoer:

```
INSERT INTO students (first_name, last_name, enrollment_date)
VALUES ('Jane', 'Smith', '2024-08-15');

INSERT INTO students (first_name, last_name, enrollment_date)
VALUES ('Emily', 'Jones', '2024-07-21');
```

2. **Kjør Kommandoene:**
   - Kjør begge INSERT-kommandoene for å legge til de nye studentene i tabellen.

Trinn 2: Lese Data (SELECT)

Hva er en SELECT-kommando?

SELECT-kommandoen brukes for å hente data fra en tabell. Du kan velge hvilke kolonner du vil se, og filtrere resultatene ved å bruke forskjellige kriterier.

Trinn 2.1: Hent Alle Data fra Tabellen

1. **Skriv en Enkel SELECT-kommando:**
   - For å hente alle rader og kolonner fra `students`-tabellen, skriv:

```
SELECT * FROM students;
```

   - **Forklaring:**
     - `*` betyr "alle kolonner".
     - `FROM students` spesifiserer at vi henter data fra `students`-tabellen.

2. **Kjør Kommandoen:**
   - Klikk på Execute-ikonet eller trykk `Ctrl + Enter` for å kjøre SELECT-kommandoen.
   - Resultatene vises i et rutenett under SQL Editor.

Trinn 2.2: Hent Spesifikke Kolonner

1. **Velg Spesifikke Kolonner:**
   - For å hente bare studentenes for- og etternavn, skriv:

```
SELECT first_name, last_name FROM students;
```

2. **Kjør Kommandoen:**
   - Klikk på Execute for å kjøre denne kommandoen. Nå vil bare fornavn og etternavn vises i resultatene.

Trinn 2.3: Filtrer Resultatene med WHERE

1. **Bruk WHERE for å Filtrere Resultater:**
   - For å hente data om en spesifikk student, for eksempel en med etternavn "Smith", skriv:

```
SELECT * FROM students WHERE last_name = 'Smith';
```

2. **Kjør Kommandoen:**
   - Klikk på Execute for å se resultatene. Bare rader hvor `last_name` er "Smith" vil bli vist.

Trinn 3: Oppdatere Data (UPDATE)

Hva er en UPDATE-kommando?

UPDATE-kommandoen brukes for å endre eksisterende data i en tabell. Du kan spesifisere hvilke rader som skal oppdateres og hvilke kolonner som skal endres.

Trinn 3.1: Oppdater Studentens Data

1. **Velg Raden du Vil Oppdatere:**
   - For å oppdatere fornavnet til studenten med `student_id` 1, skriv:

```
UPDATE students
SET first_name = 'Jonathan'
WHERE student_id = 1;
```

2. **Forklaring:**
   - `SET first_name = 'Jonathan'` : Endrer `first_name` til "Jonathan".
   - `WHERE student_id = 1` : Angir at bare raden med `student_id` lik 1 skal oppdateres.

3. **Kjør Kommandoen:**
   - Klikk på Execute for å kjøre UPDATE-kommandoen.

4. **Verifiser Oppdateringen:**
   - Bruk en SELECT-kommando (f.eks. `SELECT * FROM students WHERE student_id = 1;`) for å bekrefte at oppdateringen var vellykket.

Trinn 4: Slette Data (DELETE)

Hva er en DELETE-kommando?

DELETE-kommandoen brukes for å slette rader fra en tabell. Vær forsiktig med denne kommandoen, da slettede data ikke kan gjenopprettes.

Trinn 4.1: Slett en Rad

1. **Skriv en DELETE-kommando:**
   - For å slette studenten med `student_id` 1, skriv:

```
DELETE FROM students WHERE student_id = 1;
```

2. **Forklaring:**
   - `DELETE FROM students` : Angir at vi vil slette fra `students` -tabellen.
   - `WHERE student_id = 1` : Sørger for at bare raden med `student_id` lik 1 blir slettet.

3. **Kjør Kommandoen:**
   - Klikk på Execute for å kjøre DELETE-kommandoen.

4. **Verifiser Slettingen:**
   - Bruk en SELECT-kommando (f.eks. `SELECT * FROM students;`) for å se at raden ble slettet.

Del 4: Bruke SQL i Node.js Express API

I denne delen skal du lære hvordan du kan koble en Node.js Express-applikasjon til MySQL-serveren din og utføre SQL-spørringer fra API-et ditt. Du vil bruke moderne ES6+ syntaks for å skrive koden din, noe som gjør koden mer lesbar og effektiv.

Før du starter, må du ha Node.js og npm (Node Package Manager) installert på din maskin. Hvis du ikke har det, kan du laste ned og installere det fra Node.js offisielle nettside.

Trinn 1: Opprett en Ny Node.js Express Applikasjon

Trinn 1.1: Initialiser en Ny Node.js Applikasjon

1. **Lag en Ny Mappe for Prosjektet:**
   - Åpne terminalen (kommandolinjen) og naviger til ønsket sted, deretter skriv:

```
mkdir student-api
cd student-api
```

2. **Initialiser Node.js-prosjektet:**
   - Kjør følgende kommando for å initialisere en ny Node.js-applikasjon:

```
npm init -y
```

   - Dette vil opprette en `package.json`-fil som holder styr på avhengighetene dine.

Trinn 1.2: Installer Nødvendige Pakker

1. **Installer Express og MySQL:**
   - For å sette opp en enkel server og koble til MySQL, installer følgende pakker:

```
npm install express mysql
```

2. **Installer Nodemon for Utvikling:**
   - Nodemon er et nyttig verktøy som automatisk restarter serveren når du gjør endringer:

```
npm install --save-dev nodemon
```

3. **Oppdater `package.json` for å Bruke Nodemon:**
   - Åpne `package.json` og finn "scripts"-delen. Oppdater den til å se slik ut:

```
...

"type": "modules"
"scripts": {
  "dev": "nodemon index.js"
}

...
```

- Dette gjør at du kan starte serveren med `npm start` og automatisk oppdatere serveren når du gjør endringer.

Trinn 2: Konfigurer MySQL-tilkobling i Express

Trinn 2.1: Opprett en MySQL-Tilkobling

1. **Opprett en Fil for Serveren:**

   - Lag en ny fil kalt `index.js` i prosjektmappen din:

```
touch index.js
```

2. **Opprett en Enkel Express-server og Koble til MySQL:**

   - Åpne `index.js` og skriv følgende kode for å sette opp serveren og MySQL-tilkoblingen ved bruk av ES6+ syntaks:

```
import express from 'express';
import mysql from 'mysql';

const app = express();

// MySQL tilkoblingskonfigurasjon
const db = mysql.createConnection({
  host: 'localhost',
  user: 'app_user',
  password: 'your_password',
  database: 'student_database'
});

// Koble til MySQL
db.connect((err) => {
  if (err) {
    console.error('Feil ved tilkobling til MySQL:', err);
    return;
  }
  console.log('Koblet til MySQL database');
});

// Start serveren
app.listen(3000, () => {
  console.log('Server kjører på port 3000');
});
```

   - **Forklaring:**
     - `host` : MySQL-serverens adresse, her `localhost` for lokal maskin.
     - `user` : Brukernavnet til MySQL-brukeren du opprettet (f.eks. `app_user`).
     - `password` : Passordet til MySQL-brukeren.
     - `database` : Navnet på databasen du vil koble til (f.eks. `student_database`).

3. **Start Serveren:**

   - Kjør følgende kommando i terminalen for å starte serveren:

```
npm start
```

   - Hvis alt er konfigurert riktig, bør du se meldingen "Koblet til MySQL database" i terminalen, etterfulgt av "Server kjører på port 3000".

Trinn 3: Lag API-endepunkter for CRUD-operasjoner

Trinn 3.1: Lag et Endepunkt for å Lese Data (GET)

1. **Opprett et GET-endepunkt:**
   - I `index.js`, legg til følgende kode for å opprette et GET-endepunkt som henter alle studenter fra databasen:

```
app.get('/students', (req, res) => {
  const sql = 'SELECT * FROM students';

  db.query(sql, (err, results) => {
    if (err) {
      res.status(500).json({ error: 'Feil ved henting av data' });
      return;
    }
    res.json(results);
  });
});
```

2. **Test GET-endepunktet:**
   - Åpne en nettleser eller bruk et verktøy som Postman og naviger til `http://localhost:3000/students`.
   - Du bør få tilbake en liste over alle studentene i `students`-tabellen som JSON.

Trinn 3.2: Lag et Endepunkt for å Legge til Data (POST)

1. **Aktiver Parsing av JSON-data:**
   - Før du kan legge til data, må du sørge for at Express kan lese JSON-innhold i forespørsler:

```
app.use(express.json());
```

2. **Opprett et POST-endepunkt:**
   - Legg til følgende kode i `index.js` for å opprette et POST-endepunkt som legger til en ny student:

```
app.post('/students', (req, res) => {
  const { first_name, last_name, enrollment_date } = req.body;
  const sql = 'INSERT INTO students (first_name, last_name, enrollment_date) VALUES (?, ?, ?)';

  db.query(sql, [first_name, last_name, enrollment_date], (err, result) => {
    if (err) {
      res.status(500).json({ error: 'Feil ved innsending av data' });
      return;
    }
    res.status(201).json({ message: 'Student lagt til' });
  });
});
```

3. **Test POST-endepunktet:**
   - Bruk Postman eller lignende til å sende en POST-forespørsel til `http://localhost:3000/students` med følgende JSON-body:

```
{
  "first_name": "Alice",
  "last_name": "Johnson",
  "enrollment_date": "2024-09-01"
}
```

- Hvis alt fungerer som det skal, vil serveren returnere "Student lagt til".

Trinn 3.3: Lag et Endepunkt for å Oppdatere Data (PUT)

1. **Opprett et PUT-endepunkt:**
   - Legg til følgende kode for å opprette et PUT-endepunkt som oppdaterer en eksisterende student:

```
app.put('/students/:id', (req, res) => {
  const { first_name, last_name, enrollment_date } = req.body;
  const { id } = req.params;
  const sql = 'UPDATE students SET first_name = ?, last_name = ?, enrollment_date = ? WHERE student_id =

  db.query(sql, [first_name, last_name, enrollment_date, id], (err, result) => {
    if (err) {
      res.status(500).json({ error: 'Feil ved oppdatering av data' });
      return;
    }
    res.json({ message: 'Student oppdatert' });
  });
});
```

2. **Test PUT-endepunktet:**
   - Bruk Postman til å sende en PUT-forespørsel til `http://localhost:3000/students/1` med en JSON-body som oppdaterer dataene for student med `student_id` 1.

Trinn 3.4: Lag et Endepunkt for å Slette Data (DELETE)

1. **Opprett et DELETE-endepunkt:**
   - Legg til følgende kode for å opprette et DELETE-endepunkt som sletter en student:

```
app.delete('/students/:id', (req, res) => {
  const { id } = req.params;
  const sql = 'DELETE FROM students WHERE student_id = ?';

  db.query(sql, [id], (err, result) => {
    if (err) {
      res.status(500).json({ error: 'Feil ved sletting av data' });
      return;
    }
    res.json({ message: 'Student slettet' });
  });
});
```

2. **Test DELETE-endepunktet:**
   - Bruk Postman til å sende en DELETE-forespørsel til `http://localhost:3000/students/1` for å slette studenten med `student_id` 1.

Her er den alternative delen med bruk av **Prisma ORM** i stedet for Sequelize. Prisma er et populært og moderne ORM-verktøy som tilbyr en type-sikker og enkel måte å jobbe med databaser i Node.js-applikasjoner på.

Alternativ Del 5: Bruke Prisma ORM i Node.js Express API

I denne alternative delen skal du lære hvordan du kan bruke **Prisma ORM** for å koble en Node.js Express-applikasjon til MySQL-serveren din. Prisma tilbyr en moderne, type-sikker måte å samhandle med databasen på, som gjør koden mer robust og enkel å vedlikeholde.

Før du starter, må du ha Node.js og npm installert. Du bør også ha en MySQL-server satt opp med en database, slik som i tidligere deler av oppgaven.

Trinn 1: Installer Prisma og Nødvendige Avhengigheter

Trinn 1.1: Installer Prisma CLI og MySQL2

1. **Installer Prisma CLI og MySQL2:**
   - MySQL2 er en nødvendig avhengighet for å bruke Prisma med MySQL.

```
npm install @prisma/client
npm install --save-dev prisma
```

2. **Initialiser Prisma:**
   - Kjør følgende kommando for å initialisere Prisma i prosjektet ditt:

```
npx prisma init
```

   - Dette oppretter en `prisma`-mappe med en `schema.prisma`-fil som inneholder Prisma-skjemaet ditt, samt oppdaterer `package.json` med nødvendige skript.

Trinn 1.2: Konfigurer Prisma-skjema

1. **Konfigurer `schema.prisma`:**
   - Åpne `prisma/schema.prisma` og konfigurer det slik at det kobler til MySQL-databasen:

```
datasource db {
  provider = "mysql"
  url      = env("DATABASE_URL")
}

generator client {
  provider = "prisma-client-js"
}

model Student {
  id             Int      @id @default(autoincrement())
  first_name     String
  last_name      String
  enrollment_date DateTime
}
```

   - **Forklaring:**
     - `datasource`: Definerer datakilden, her en MySQL-database. `url` leser databaseforbindelsen fra en miljøvariabel (`.env`-filen).
     - `model`: Definerer en modell som representerer `students`-tabellen i databasen med felter for `id`, `first_name`, `last_name`, og `enrollment_date`.

2. **Oppdater `.env`-filen:**

- Åpne `.env`-filen i roten av prosjektet og sett `DATABASE_URL` til å peke på din lokale MySQL-database:

```
DATABASE_URL="mysql://app_user:your_password@localhost:3306/student_database"
```

Trinn 2: Generer Prisma Client og Migrer Database

Trinn 2.1: Generer Prisma Client

1. **Generer Prisma Client:**

   - Kjør følgende kommando for å generere Prisma Client basert på skjemaet:

```
npx prisma generate
```

   - Dette genererer TypeScript- eller JavaScript-kode for å jobbe med datamodellene dine.

Trinn 2.2: Kjør Migrasjon for å Opprette Tabellen

1. **Kjør første migrasjon:**

   - Lag en migrasjon som oppretter `students`-tabellen i databasen:

```
npx prisma migrate dev --name init
```

   - **Forklaring:**
     - `migrate dev`: Brukes til å kjøre migrasjoner under utvikling.
     - `--name init`: Gir migrasjonen et navn (her `init` for initialisering).

Trinn 3: Lag API-endepunkter ved hjelp av Prisma

Trinn 3.1: Sett opp Prisma Client i Express

1. **Opprett Prisma Client i `index.js`:**

   - Åpne `index.js` og legg til følgende kode for å sette opp Prisma Client:

```
import { PrismaClient } from '@prisma/client';

const prisma = new PrismaClient();

// Start serveren
app.listen(3000, () => {
  console.log('Server kjører på port 3000');
});
```

Trinn 3.2: Lag et Endepunkt for å Lese Data (GET)

1. **Opprett et GET-endepunkt:**

   - I `index.js`, legg til følgende kode for å opprette et GET-endepunkt som henter alle studenter fra databasen:

```
app.get('/students', async (req, res) => {
  try {
    const students = await prisma.student.findMany();
    res.json(students);
```

```
  } catch (error) {
    res.status(500).json({ error: 'Feil ved henting av data' });
  }
});
```

Trinn 3.3: Lag et Endepunkt for å Legge til Data (POST)

1. **Opprett et POST-endepunkt:**
   - Legg til følgende kode i `index.js` for å opprette et POST-endepunkt som legger til en ny student:

```
app.post('/students', async (req, res) => {
  try {
    const { first_name, last_name, enrollment_date } = req.body;
    const newStudent = await prisma.student.create({
      data: { first_name, last_name, enrollment_date }
    });
    res.status(201).json(newStudent);
  } catch (error) {
    res.status(500).json({ error: 'Feil ved innsending av data' });
  }
});
```

Trinn 3.4: Lag et Endepunkt for å Oppdatere Data (PUT)

1. **Opprett et PUT-endepunkt:**
   - Legg til følgende kode for å opprette et PUT-endepunkt som oppdaterer en eksisterende student:

```
app.put('/students/:id', async (req, res) => {
  try {
    const { id } = req.params;
    const { first_name, last_name, enrollment_date } = req.body;
    const updatedStudent = await prisma.student.update({
      where: { id: parseInt(id) },
      data: { first_name, last_name, enrollment_date }
    });
    res.json(updatedStudent);
  } catch (error) {
    res.status(500).json({ error: 'Feil ved oppdatering av data' });
  }
});
```

Trinn 3.5: Lag et Endepunkt for å Slette Data (DELETE)

1. **Opprett et DELETE-endepunkt:**
   - Legg til følgende kode for å opprette et DELETE-endepunkt som sletter en student:

```
app.delete('/students/:id', async (req, res) => {
  try {
    const { id } = req.params;
    await prisma.student.delete({
      where: { id: parseInt(id) }
    });
    res.json({ message: 'Student slettet' });
  } catch (error) {
    res.status(500).json({ error: 'Feil ved sletting av data' });
  }
});
```