Rapport - Databaseprosjekt

Laget av: Joel Constantinos, Kim-Iver Blindheimsvik, Julius Stensen

Beskrivelse av applikasjon:
Applikasjonen funker ved at brukeren må kjøre main.py-filen gjennom terminalen. Når bruker gjør det, så får valget mellom å lage en bruker eller å bruke en eksisterende bruker. Hvis terminalbrukeren lager en bruker, så blir den lagt til databasen, og terminalbrukeren kan velge å bruke den videre i programmet. Etter terminalbrukeren velger bruker, så får den valget mellom å velge de fem forskjellige brukerhistoriene. Ved å velge brukerhistorie 2-5, så printes det resultatene av SQL-spørringene ut i terminalen og programmet avsluttes. Ved å velge brukerhistorie 1, kan terminalbrukeren legge inn en kaffesmaking ved å oppgi kaffe, brennerinavn, smaksnotat og poengscore til terminalen. Programmet blir så avsluttet ved at kaffesmakingen/kaffeanmeldelsen legges til databasen.

Sqlite-beskrivelse i python:
For å bruke sqlite i python må vi importere sqlite3 biblioteket som gjøres i linje 1 i figuren nedenfor. Sql databasen er kobles til python ved bruk av kommandoen i linje 2 og linje 3. Videre, ser vi i linje 4 hvordan man utfører handlinger som oprettelse av tabeller, innsetting av data, og spørring for å hente ut data. I midten av de seks anførselstegnene skriver man bare vanlig sqlite kode. Til slutt ser vi i linje 5 hvordan man forsikrer at hendelsene blir utført på databasen før man lukker tilkoblingen, og linje 6 blir tilkoblingen til databasen fjernet.

```
1  import sqlite3
2  con = sqlite3.connect("kaffe.db")
3  cursor = con.cursor()
4  cursor.execute(""" """)
5  con.commit()
6  con.close()
```

Nedenfor har vi besvart oppgave 2.b) og 2.c) for hver av brukerhistoriene. Ved brukerhistorie 1 er deloppgavene besvart ved en tekstbeskrivelse av hvordan app interfacet fungerer. Mens brukshistorie 2 til brukerhistorie 5 viser antagelser tatt i spørringene og outputen av dem til terminalen fra python.

Brukerhistorie 1)

Brukerhistorie 1 er implementert på følgende måte. Først selekteres det brukernavn- og passordkolonner fra bruker-entiteten i databasen gjennom spørringen: SELECT epostadresse, passord FROM bruker, slik at brukeren kan skrive inn i terminalen hvilken bruker i databasen han ønsker å bruke. Denne epostadressen til brukeren lagres så i en epostadresse-variabel som brukes for å identifisere kaffeanmeldelsen til brukeren senere.

De ulike kaffe- og brennerinavnene lagres så i en liste med flere tupler som inneholder verdiene gjennom den følgende SQL-spørring: SELECT brennerinavn, navn FROM kaffe, og cursor.fetchall() på en ny linje. Terminalbrukeren får så skrevet ut alle kaffe og brennerikombinasjonene i listen med tuplene gjennom funksjonen print_kaffe_brennerier(). Terminalbrukeren må så skrive inn en gyldig kombinasjon av kaffe og brenneri når han får opp spørsmål gjennom to innputsetninger om hvilken kaffe og brenneri det er. Så bes terminalbrukeren om å skrive inn en input angående hvilken poengscore kaffeen får. KaffeID bestemmes så gjennom spørringen: “SELECT kaffeID FROM kaffe WHERE kaffe.navn = ? AND kaffe.brennerinavn = ?”, [navn, brenneri] og cursor.fetchall() som lagrer den i en tuppel. Her selekteres så kaffeID gjennom å skrive ut kaffeID[0][0] temp for å få et heltall.

smaksDato er definert manuelt som en variabel i funksjonen. Disse fem variablene settes så inn i databasen gjennom statementen “INSERT INTO kaffesmaking(epostadresse, smaksDato, kaffeID, poengscore, smaksNotater) VALUES (?, ?, ?, ?, ?)”, (epostadresse, smaksDato, kaffeID, poeng, smaksnotat)). Resten av variablene fylles inn gjennom fremmednøkler som er i scriptet. Dermed så er det mulig for en terminalbruker å legge inn en kaffesmaking/kaffeanmeldelse som baserer seg på de inputene som er kravet i brukerhistorie 1. I tillegg, så er brukerhistorien med variablene som er beskrevet i brukerhistorie 1 lagt inn manuelt i databasen gjennom funksjonen kaffeData5().

Brukerhistorie 2)

For å forklare hvordan appen oppfyller brukehistorie 2 har vi hentet ut en liste over brukerens fullenavn og antall unike kaffesmakinger en bruker har smakt i år 2022. Antagelsen vi har tatt er at appen oppdaterer automatisk hvilket år det er, slik at listen holder seg oppdatert.

Sql query:

```
SELECT bruker.navn as Brukerens_Fulle_Navn,
COUNT(DISTINCT kaffesmaking.kaffeID) AS Antall_Smaksnotater
FROM kaffesmaking
INNER JOIN bruker
ON bruker.epostadresse = kaffesmaking.epostadresse
Where kaffesmaking.smaksDato LIKE '%2022%'
GROUP BY bruker.navn
ORDER BY Antall_Smaksnotater DESC
```

Output fra spørring i terminalen

Fulle navn	Antall smaksnotater
Kim-Iver Blindheimsvik	2
Torleif Brandtzæg	1
Julius Schjetne	1

Brukerhistorie 3)

Her har vi hentet ut en liste over brennerinavn, kaffenavn, kilopris og gjennomsnittsscoren en kaffe har fått. Output er en liste over hvilke kaffe som gir bruker mest for pengene i synkende rekkefølge. Vi antar at poengskalaen går fra 0 til 10, og at brukere ikke legger inn verdier utenfor dette vinduet.

Sql query:

```
SELECT kaffe.brennerinavn as Brennerinavn, kaffe.navn AS Kaffenavn,
ROUND(kaffe.kilopris, 1) AS Kilopris,
ROUND(AVG(poengscore),1) AS Gjennomsnittsscore
FROM kaffe
INNER JOIN kaffesmaking ON kaffesmaking.kaffeID = kaffe.kaffeID
GROUP BY kaffesmaking.kaffeID
ORDER BY kaffe.kilopris/Gjennomsnittsscore ASC
```

Output fra spørring i terminalen:

Brennerinavn	Kaffenavn	KiloprisNOK	Gjennomsnitlig score fra brukere
Brenneri B	Nyår kaffe 2021	300.0	9.0
Jacobsens & Svart	Vinterkaffe 2022	600.0	9.0
Brenneri A	Julekaffe 2020	500.0	7.0
Brenneri D	Sommerkaffe 2018	900.0	8.5
Brenneri C	Sommerkaffe 2021	900.0	5.7
Brenneri E	Sommerkaffe 2017	1000.0	6.0

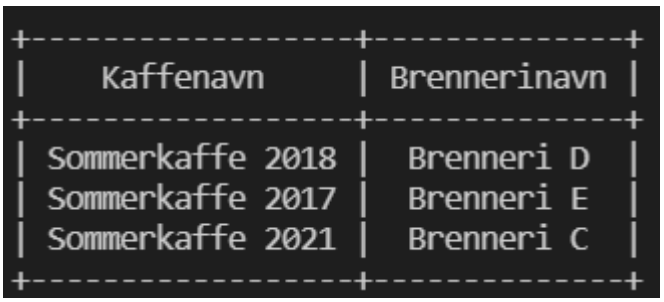
Brukerhistorie 4)

Spørringen under tillater brukeren å få en liste bestående av kaffenavn og brennerinavn beskrevet med ordet floral. Hvor hver unike kaffe og brenneri kombinasjon vil være en del av listen. Alle radene i listen har blitt beskrevet med ordet floral av en bruker, eller et brenneri, eller både en bruker og et brenneri.
Antagelsen vi har tatt her er at to kaffer kan ha samme navn men komme fra ulik brennerier.

Sql query:

```
SELECT DISTINCT kaffe.navn AS Kaffenavn, kaffe.brennerinavn as Brennerinavn
FROM kaffe
INNER JOIN kaffesmaking ON kaffesmaking.kaffeID = kaffe.kaffeID
WHERE kaffesmaking.smaksNotater LIKE '%floral%' OR kaffe.beskrivelse LIKE
'%floral%'
```

Output fra spørring i terminalen:



Kaffenavn	Brennerinavn
Sommerkaffe 2018	Brenneri D
Sommerkaffe 2017	Brenneri E
Sommerkaffe 2021	Brenneri C

Brukerhistorie 5)

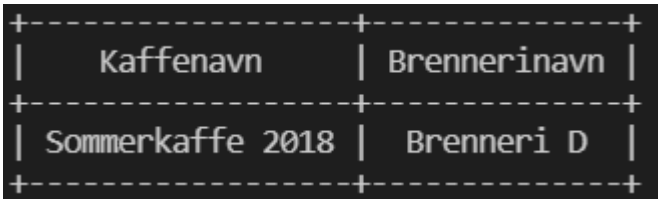
Her har vi hentet ut en liste over kaffenavn og brennerinavn til kaffe som er laget i Colombia eller Rwanda, og som ikke er vasket. Her antar vi at ordet “Vasket” inngår i navnet til alle foredlingsmetoder hvor kaffebønner vaskes.

Sql query:

```
SELECT kaffe.navn AS Kaffenavn, kaffe.brennerinavn AS Brennerinavn
FROM kaffe
```

```
INNER JOIN kaffebønneParti ON kaffebønneParti.partiID = kaffe.partiID
INNER JOIN gård ON kaffebønneParti.gårdID = gård.gårdID
WHERE (gård.land = 'Rwanda' OR gård.land = 'Colombia')
AND kaffebønneParti.foredlingsmetode NOT LIKE '%Vasket%'
```

Output fra spørring i terminalen:



Kaffenavn	Brennerinavn
Sommerkaffe 2018	Brenneri D