Øving 4 - SQL del 2

Oppgave 1 - SQL inkl. VIEW

> Bruk dette scriptet levInfo_mysql.txt for å opprette tabeller med eksempeldata. Gjør deg kjent med databasen. Merk at vi ikke har NULL-verdier i denne databasen.

1.a

> List ut all informasjon (ordrehode og ordredetalj) om ordrer for leverandør nr 44.

Spørring:

```sql
SELECT
    ordrehode.*,
    ordredetalj.*
FROM
    ordrehode
INNER JOIN
    ordredetalj ON ordrehode.ordrenr = ordredetalj.ordrenr
WHERE
    ordrehode.levnr = 44;
```

Fra tabellen ordrehode som primærtabell, velges alle kolonner i begge tabellene ved bruk av et "wildcard", `*`.

Deretter flettes tabellene sammen om deres felles rader og kolonner, basert på `ordrenr`, og filtrerer til slutt resultatet på leverandør nr. 44.

Resultat:

ordrenr	dato	levnr	status	delnr	kvantum
13	1986-09-13	44	p	51173	20
14	1986-12-17	44	p	201	100
14	1986-12-17	44	p	202	100
14	1986-12-17	44	p	51173	30
15	1987-01-03	44	p	201	100
15	1987-01-03	44	p	202	100

1.b

> Finn navn og by ("LevBy") for leverandører som kan levere del nummer 1.

Spørring:

```sql
SELECT
    levinfo.navn,
    levinfo.levby
FROM
    levinfo
INNER JOIN
    prisinfo ON levinfo.levnr = prisinfo.levnr
WHERE
        prisinfo.delnr = 1;
```

Resultat:

navn	levby
Kontorekspressen AS	Oslo
Kontorutstyr AS	Ås

1.c

> Finn nummer, navn og pris for den leverandør som kan levere del nummer 201 til billigst pris.

Spørring:

```sql
SELECT
    levinfo.navn,
    levinfo.levby,
    prisinfo.pris
FROM
    levinfo
INNER JOIN
    prisinfo ON levinfo.levnr = prisinfo.levnr
WHERE
    prisinfo.delnr = 201
ORDER BY pris ASC
LIMIT 1;

/* Alternativ til LIMIT 1;

AND prisinfo.pris =(
  SELECT MIN(pris)
  FROM prisinfo
```

```
  WHERE delnr = 201);

*/
```

Resultat

navn	levby	pris
Billig og Bra AS	Oslo	1.6

1.d

> Lag fullstendig oversikt over ordre nr 16, med ordrenr, dato, delnr, beskrivelse, kvantum, (enhets-)pris og beregnet beløp (=pris*kvantum).

Spørring:

```
SELECT
    ordredetalj.delnr, kvantum,
    ordrehode.ordrenr, dato, levnr, status,
    delinfo.beskrivelse
FROM
    ordrehode
INNER JOIN
    ordredetalj ON ordrehode.ordrenr = ordredetalj.ordrenr
INNER JOIN
    delinfo on ordredetalj.delnr = delinfo.delnr
WHERE
    ordredetalj.ordrenr = 16;
```

Resultat:

delnr	kvantum	ordrenr	dato	levnr	status	beskrivelse
201	50	16	1987-01-31	6	c	Svarte kulepenner
202	50	16	1987-01-31	6	c	Blå kulepenner
1909	10	16	1987-01-31	6	c	Skriveunderlag
51173	20	16	1987-01-31	6	c	Binders

Etter av oppgaven blir lest skikkelig, tas også **beregnet beløp** med:

```
SELECT
    ordredetalj.delnr, kvantum,
    ordrehode.ordrenr, dato,
```

```
    delinfo.beskrivelse,
    prisinfo.pris AS enhetspris,
    ordredetalj.kvantum * prisinfo.pris AS beregnet_beløp
FROM
    ordrehode
INNER JOIN
    ordredetalj ON ordrehode.ordrenr = ordredetalj.ordrenr
INNER JOIN
    delinfo ON ordredetalj.delnr = delinfo.delnr
INNER JOIN
    prisinfo ON delinfo.delnr = prisinfo.delnr AND ordrehode.levnr =
prisinfo.levnr
WHERE
    ordrehode.ordrenr = 16;
```

Resultat:

delnr	kvantum	ordrenr	dato	beskrivelse	enhetspris	beregnet_beløp
201	50	16	1987-01-31	Svarte kulepenner	1.9	95
202	50	16	1987-01-31	Blå kulepenner	6.5	325
1909	10	16	1987-01-31	Skriveunderlag	0.85	8.5
51173	20	16	1987-01-31	Binders	0.57	11.399999999999999

1.e

> Finn delnummer og leverandørnummer for deler som har en pris som er høyere enn prisen for del med katalognr X7770.

Denne er litt tricky! Her må man til med en "underspørring" (sub query). Først finner vi prisen for den aktuelle delen, og deretter bruke denne prisen til å filtrere den ytre hovedspørringen.

Spørring:

```
SELECT
    delnr,
    levnr
FROM
    prisinfo
WHERE
    pris > (SELECT pris FROM prisinfo WHERE katalognr = 'X7770');
```

Resultat:

delnr	levnr
1	6
1	9
3	6
3	9
3	82
4	6
4	82
202	6
51200	6

1.f-i

> i) Tenk deg at tabellen levinfo skal deles i to. Sammenhengen mellom by og fylke skal tas ut av tabellen. Det er unødvendig å lagre fylkestilhørigheten for hver forekomst av by. Lag én ny tabell som inneholder byer og fylker. Fyll denne med data fra levinfo. Lag også en tabell som er lik levinfo unntatt kolonnen Fylke. (Denne splittingen av tabellen levinfo gjelder bare i denne oppgaven. I resten av oppgavesettet antar du at du har den opprinnelige levinfo-tabellen.)

Input

```
CREATE TABLE byFylke (
    byfylke_id INT auto_increment PRIMARY KEY,
    levby VARCHAR(30) NOT NULL,
    fylke VARCHAR(30) NOT NULL,
    UNIQUE (levby, fylke)
) ENGINE=INNODB;

INSERT INTO byFylke (levby, fylke)
SELECT DISTINCT levby, fylke FROM levinfo;

CREATE TABLE levinfoUtenFylke (
    levnr INT PRIMARY KEY,
    navn VARCHAR(30) NOT NULL,
    adresse VARCHAR(30) NOT NULL,
    postnr INT NOT NULL,
    levby VARCHAR(30) NOT NULL,
```

```
    FOREIGN KEY (levby) REFERENCES byFylke(levby)
) ENGINE=INNODB;

INSERT INTO levinfoUtenFylke (levnr, navn, adresse, postnr, levby)
SELECT levnr, navn, adresse, postnr, levby FROM levinfo;
```

byFylke-tabellen:

byfylke_id	levby	fylke
3	Ål	Telemark
2	Ås	Østfold
1	Oslo	Oslo
4	Trondheim	S-Trøndelag

levinfoUtenFylke-tabellen:

levnr	navn	adresse	postnr	levby
6	Kontorekspressen AS	Skolegata 3	1234	Oslo
9	Kontorutstyr AS	Villa Villekulla	1456	Ås
12	Mister Office AS	Storgt 56	1456	Ås
44	Billig og Bra AS	Aveny 56	1222	Oslo
81	Kontorbutikken AS	Gjennomveien 78	3345	Ål
82	Kontordata AS	Åsveien 178	7023	Trondheim

1.f-ii

> ii) Lag en virtuell tabell (view) slik at brukerne i størst mulig grad kan jobbe på samme måte mot de to nye tabellene som den gamle. Prøv ulike kommandoer mot tabellen (select, update, delete, insert). Hvilke begrensninger, hvis noen, har brukerne i forhold til tidligere?

Opprette VIEW

```sql
CREATE VIEW virtuellLevinfo AS
SELECT
    levinfoUtenFylke.levnr,
    levinfoUtenFylke.navn,
    levinfoUtenFylke.adresse,
    levinfoUtenFylke.postnr,
    levinfoUtenFylke.levby,
    byFylke.fylke
FROM
    levinfoUtenFylke
INNER JOIN
    byFylke ON levinfoUtenFylke.levby = byFylke.levby;
```

Resultat:

levnr	navn	adresse	postnr	levby	fylke
6	Kontorekspressen AS	Skolegata 3	1234	Oslo	Oslo
9	Kontorutstyr AS	Villa Villekulla	1456	Ås	Østfold
12	Mister Office AS	Storgt 56	1456	Ås	Østfold
44	Billig og Bra AS	Aveny 56	1222	Oslo	Oslo
81	Kontorbutikken AS	Gjennomveien 78	3345	Ål	Telemark
82	Kontordata AS	Åsveien 178	7023	Trondheim	S-Trøndelag

Begrensninger

- `INSERT`: En virtuell tabelln (`VIEW`) tillater ikke å sette inn data direkte. De må settes inn i de underliggende tabellene som brukes for å konstruere den virtuelle tabellen.
- `UPDATE`: Som med `INSERT`, må `UPDATE` utføres på de underliggende tabellene.
- `DELETE`: Samme som `INSERT` og `UPDATE`. Må utføres på de underliggende tabellene.

Altså er ikke et `VIEW` egnet for å manipulere tabeller, altså endre dataene. Dette må gjøres i tabellene som brukes for å konstruere et `VIEW`. Et `VIEW` er hovedsaklig ment for å hente ut (`SELECT`) data og vise det i egne "skrivebeskyttede", "read-only" tabeller.

1.g

> Anta at en vurderer å slette opplysningene om de leverandørene som ikke er representert i Prisinfo-tabellen. Finn ut hvilke byer en i tilfelle ikke får leverandør i. (Du skal ikke utføre slettingen.) (Tips: Svaret skal bli kun én by, "Ål".)

For å finne hvilke byer man ikke får leverandør i, dersom man sletter de som ikke er representert i Prisinfo-tabellen kan følgende spørring kjøres:

```
SELECT
    levinfo.levby
FROM
    levinfo
LEFT JOIN
    prisinfo ON levinfo.levnr = prisinfo.levnr
WHERE
    prisinfo.levnr IS NULL;
```

Resultat:

levby
Ås
Ål

Dette stemmer ikke med "tipset" oppgitt i oppgaven. Prøver en ny spørring, med en sub-query for å filtrere ut mer:

```
SELECT DISTINCT
    levinfo.levby
FROM
    levinfo
WHERE
    levinfo.levnr NOT IN (SELECT prisinfo.levnr FROM prisinfo INNER JOIN levinfo
ON levinfo.levnr = prisinfo.levnr);
```

Resultat:

levby
Ås
Ål

Igjen kommer det ut to byer.

Utfører en ny spørring for å sjekke tabellen manuelt for ev. feil:

```sql
SELECT DISTINCT
    levinfo.levby,
    levinfo.levnr,
    prisinfo.levnr
FROM
    levinfo
LEFT JOIN
    prisinfo ON levinfo.levnr = prisinfo.levnr
```

Resultat:

levby	levnr	levnr
Oslo	6	6
Ås	9	9
Ås	12	null
Oslo	44	44
Ål	81	null
Trondheim	82	82

Ser her at både Ås og Ål har `null` i `levnr`, altså finnes det ingen leverandører i disse byene.

Konklusjon: Enten er tipset i oppgaven feil, eller så har jeg gjort noe feil.

Løsning:

```sql
SELECT DISTINCT
    levinfo.levby
FROM
    levinfo
WHERE
        levinfo.levby NOT IN (SELECT levinfo.levby FROM prisinfo INNER JOIN
levinfo ON levinfo.levnr = prisinfo.levnr);
```

1.h

> Finn leverandørnummer for den leverandør som kan levere ordre nr 18 til lavest totale beløp (vanskelig).
>
> Hint: Løs oppgaven i tre steg:
>
> - Lag en virtuell tabell (view) som viser hvem som kan levere hele eller deler av ordren.
> - Fra denne velger du så ut de leverandørene som kan levere like mange deler til ordren som ordren krever. Det vil si, kan levere hele ordren.

> - Til slutt finner du ut hvem som leverer billigst.
> - Svaret skal bli at leverandør 6 kan levere ordren for 6798 kroner.

Steg 1

Lager en `VIEW` med leverandører som kan levere hele ordren, `LeverandørerOrdre18`.

```sql
CREATE VIEW LeverandørerOrdre18 AS
SELECT
    prisinfo.levnr,
    ordredetalj.delnr,
    ordredetalj.kvantum,
    prisinfo.pris * ordredetalj.kvantum AS totalpris
FROM
    prisinfo
INNER JOIN
    ordredetalj ON prisinfo.delnr = ordredetalj.delnr
WHERE
    ordredetalj.ordrenr = 18;
```

Resultatet er en tabell over leverandørene som kan levere ønsket antall deler av delenummer 3 (2 stk) og 4 (8 stk), som vi kan finne manuelt ved å sjekke tabellen `ordredetalj` og finne ordrenummer 18.

levnr	delnr	kvantum	totalpris
6	3	2	2398
9	3	2	2100
82	3	2	2598
6	4	8	4400
82	4	8	7192

Steg 2

Velger leverandører som kan levere **hele** ordren. Teller opp antallet unike deler i ordren og filterere ut leverandørene som kan levere dette antallet deler.

```sql
SELECT
    levnr
FROM
    LeverandørerOrdre18
GROUP BY
    levnr
HAVING
    COUNT(DISTINCT delnr) = (SELECT COUNT(DISTINCT delnr) FROM ordredetalj WHERE ordrenr = 18);
```

Resultat:

levnr
6
82

Steg 3

Finner den som kan levere billigst:

```sql
SELECT
    levnr,
    SUM(totalpris) AS totalbeløp
FROM
    LeverandørerOrdre18
WHERE
        levnr IN ( /*Her putter vi inn spørringen/resultatet fra steg 2 som en sub-query*/
        SELECT
            levnr
        FROM
            LeverandørerOrdre18
        GROUP BY
            levnr
        HAVING
            COUNT(DISTINCT delnr) = (SELECT COUNT(DISTINCT delnr) FROM ordredetalj WHERE ordrenr = 18)
    )
GROUP BY
    levnr
ORDER BY
    totalbeløp ASC
LIMIT 1;
```

Resultat:

levnr	totalbeløp
6	6798

Resultatet er i henhold til det oppgaven oppgir som riktig løsning.

Oppgave 2 - SQL med NULL-verdier

> Bruk Bok-databasen (bok-script-mysql.txt) lagt ut tidligere i faget. Gå gjennom datasettet og finn ut hvor det ligger NULL-verdier.

2.a

> Sett opp en SELECT-setning som er UNION mellom alle forlag med Oslo-nummer (telefonnummer begynner med 2) og alle som ikke er Oslo-nummer. Får du med forlaget med NULL-verdi på telefonnummer? Hvis ikke, utvid unionen med en mengde til.

Spørring:

```sql
/*
Del 1: Velger alle forlag med telefonnummer som begynner med 2 (Oslo)
*/
(SELECT * FROM forlag WHERE telefon LIKE '2%')

UNION

/*
Del 2: Velger alle forlag med telefonnummer som ikke begynner med 2
*/
(SELECT * FROM forlag WHERE telefon NOT LIKE '2%' AND telefon IS NOT NULL)
/* Trenger egenltig ikke IS NOT NULL */
UNION

/*
Del 3: Velger alle forlag med NULL på telefonnummer
*/
(SELECT * FROM forlag WHERE telefon IS NULL);
/* Her må vi ha IS NULL
Kunne også brukt SELECT *
*/
```

Spørringen kan kjøres i tre steg, uten `UNION`, eller kombinert med `UNION`.

Resultater

Del 1 - Oslo-nummer

forlag_id	forlag_navn	adresse	telefon
2	Gyldendal	Oslo	22220000
3	Cappelen	Oslo	22200000
4	Universitetsforlaget	Oslo	23230000
5	Aschehaug	Oslo	22000000
6	Oktober	Oslo	22002200
7	Tiden	Oslo	22232223

Del 2 - Oslo-nummer og ikke-NULL

forlag_id	forlag_navn	adresse	telefon
2	Gyldendal	Oslo	22220000
3	Cappelen	Oslo	22200000
4	Universitetsforlaget	Oslo	23230000
5	Aschehaug	Oslo	22000000
6	Oktober	Oslo	22002200
7	Tiden	Oslo	22232223
1	Tapir	Trondheim	73590000

Del 3 - Alle

forlag_id	forlag_navn	adresse	telefon
2	Gyldendal	Oslo	22220000
3	Cappelen	Oslo	22200000
4	Universitetsforlaget	Oslo	23230000
5	Aschehaug	Oslo	22000000
6	Oktober	Oslo	22002200
7	Tiden	Oslo	22232223
1	Tapir	Trondheim	73590000
8	Harper Collins	USA	null

2.b

> Sett opp SQL-setninger som finner gjennomsnittlig alder på forfattere der fødselsåret er oppgitt. For forfattere der dødsåret ikke er oppgitt, skal du kun ta med de som er født etter 1900. Tips for å få ut året i år:
> MySQL: SELECT YEAR(CURRENT_DATE) FROM ... hvilken tabell som helst ...

Spørring:

```sql
/* Finner gjennomsnittsalderen for forfattere der fødselsåret er oppgitt */
WITH AuthorAge AS (
    /* Forfattere der både fødselsår og dødsår er oppgitt */
    SELECT (dod_aar - fode_aar) AS age
    FROM forfatter
    WHERE fode_aar IS NOT NULL AND dod_aar IS NOT NULL

    UNION ALL

    /* Forfattere der bare fødselsår er oppgitt, og som er født etter 1900 */
    SELECT (YEAR(CURRENT_DATE) - fode_aar) AS age
    FROM forfatter
    WHERE fode_aar IS NOT NULL AND dod_aar IS NULL AND fode_aar > 1900
)

/* Regner ut gjennomsnittsalderen for forfatterne */
SELECT AVG(age) AS AverageAge
FROM AuthorAge;
```

Definerer et midlertidig resultatsett, CTE (Common Table Expression) og kaller den `AuthorAge`.

I første `SELECT`-setning, beregnes alderen til forfattere som har oppgitt både fødsels- og dødsår (altså de er døde).

I andre `SELECT`-setning beregnes alderen til forfattere som fortsatt er i live, altså ikke har et dødsår. Dette gjøres ved å bruke `YEAR(CURRENT_DATE)`, som dermed bruker nåværende kalenderår for å beregne alderen.

Utenfor CTE-settet beregnes så gjennomsnittsalderen for de aktuelle forfatterne ved å bruke `AVG`-funksjonen (average, gjennomsnitt) på kolonnen `age` (alder) fra CTE-tabellen `AuthorAge`.

Resultat:

AverageAge
68.6000

2.c

> Sett opp SQL-setninger som finner hvor stor andel av forfatterne som ble med i beregningene under b).

Konstruerer to CTE-tabeller:

- `EligibleAuthors`: Teller opp antallet forfattere som oppfyller kriteriene ("eligible") satt i oppgave 2b, basert på årstall født og død.
- `TotalAuthors`: Teller opp det totale antallet forfattere i datasettet.

Beregner deretter andelen forfattere som oppfyller kravene fra 2b, ved å dele antall gunstige (`EligibleCount`) på antall mulige (`TotalCount`). Ganger dette med 100 for å få svaret i prosent, og legger det i en egen kolonne/tabell.

Spørring:

```sql
WITH
    /* Finner antall forfattere som oppfyller kriteriene fra 2b */
    EligibleAuthors AS (
        SELECT COUNT(*) AS EligibleCount
        FROM forfatter
        WHERE (fode_aar IS NOT NULL AND dod_aar IS NOT NULL)
           OR (fode_aar IS NOT NULL AND dod_aar IS NULL AND fode_aar > 1900)
    ),
    /* Finner totalt antall forfattere */
    TotalAuthors AS (
        SELECT COUNT(*) AS TotalCount
        FROM forfatter
    )
/* Beregner andelen av forfattere som oppfyller kriteriene fra 2b*/
SELECT
        (CAST(EligibleCount AS DECIMAL) / TotalCount) * 100 AS
EligibleAuthorPercentage
FROM
    EligibleAuthors, TotalAuthors;

```

Resultat:

EligibleAuthorPercentage
41.6667