

Oblig 1

Anders P. Åsbø

(Dated: October 4, 2022)

Til denne obligen brukte jeg en database som kjørte på Wampserver med MySQL 8.0.27, men jeg brukte JetBrains DataGrip til å faktisk skrive, samt kjøre SQL-koden og interagere med databasen. Kodeutklippene i dette dokumentet er alle hentet fra terminalen, slik at de inkluderer resultat-meldinger fra da de ble kjørt. I tillegg inkluderer de fleste kodeutklippene en ekstra spørring for å hente relevant info fra tabeller for å se endringene. En SQL-fil med all koden brukt i denne besvarelsen finner du her: https://github.com/FunkMarvel/DatabaserOblig1/blob/37b82b2bb52ebbd00d276b7ced98bb2fcd15f9d1/src/oblig1_queries.sql

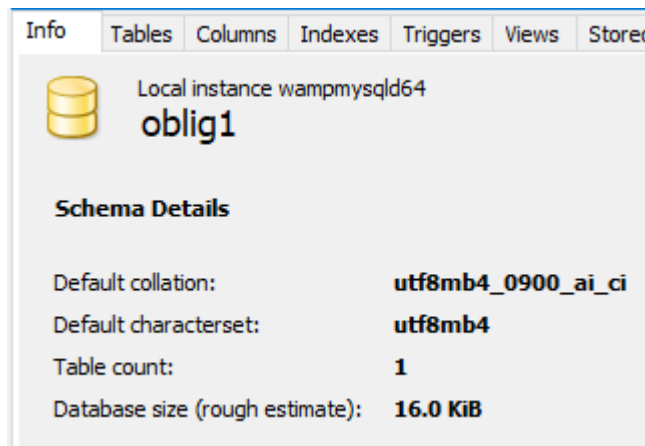
OPPGAVE 1

a)

For å lage databasen "Oblig1" bruker jeg sql setningen

```
oblig1> CREATE SCHEMA Oblig1
[2022-09-11 16:51:14] 1 row affected in 9 ms
```

Som resulterer i




b)

For å lage tabellen "Film", så bruker jeg følgende sql:

```
oblig1> CREATE TABLE Oblig1.Film (
  FNr INTEGER UNSIGNED NOT NULL,
  Tittel VARCHAR(40) NOT NULL,
  År SMALLINT UNSIGNED,
  Land VARCHAR(40),
  Sjanger VARCHAR(20),
  Alder TINYINT UNSIGNED,
  Tid SMALLINT UNSIGNED,
  Pris DECIMAL(5, 2),
  PRIMARY KEY (FNr)
)
[2022-09-11 17:25:37] completed in 19 ms
```

som resulterer i

Info	Columns	Indexes	Triggers	Foreign keys	Partitions	Grants	DDL
 Local instance wampmysqld64 oblig1.film							
Table Details							
Engine:	InnoDB						
Row format:	Dynamic						
Column count:	8						
Table rows:	0						
AVG row length:	0						
Data length:	16.0 KiB						
Index length:	0.0 bytes						
Max data length:	0.0 bytes						
Data free:	0.0 bytes						
Table size (estimate):	16.0 KiB						
File format:							
Data path:	C:\ProgramData\MySQL\MySQL Server 8.0\Data\oblig1\film.ibd						
Update time:							
Create time:	2022-09-11 17:59:03						
Auto increment:							
Table collation:	utf8mb4_0900_ai_ci						
Create options:							
Comment:							

c)

Jeg legger til dataene fra tabellen i boken med følgende sql:

```
oblig1> INSERT INTO Oblig1.Film
VALUES (1, 'Casablanca', 1942, 'USA', 'Drama', 15, 102, 149.00),
(2, 'Fort Apache', 1948, 'USA', 'Western', 15, 127, NULL),
(3, 'Apocalypse Now', 1979, 'USA', 'Action', 18, 155, 123.00),
(4, 'Streets of Fire', 1984, 'USA', 'Action', 15, 93, NULL),
(5, 'High Noon', 1952, 'USA', 'Western', 15, 85, 123.00),
(6, 'Cinema Paradiso', 1988, 'Italia', 'Komedie', 11, 123, NULL),
(7, 'Asterix hos britene', 1988, 'Frankrike', 'Tegnefilm', 7, 78, 149.00),
(8, 'Veiviseren', 1987, 'Norge', 'Action', 15, 96, 87.00),
(9, 'Salmer fra kjøkkenet', 2002, 'Norge', 'Komedie', 7, 80, 149.00),
(10, 'Anastasia', 1997, 'USA', 'Tegnefilm', 7, 94, 123.00),
(11, 'La Grande bouffe', 1973, 'Frankrike', 'Drama', 15, 129, 87.00),
(12, 'Blues Brothers 2000', 1998, 'USA', 'Komedie', 11, 124, 135.00),
(13, 'Beatles: Help', 1965, 'Storbritannia', 'Musikk', 11, 144, NULL)
[2022-09-11 18:13:34] 13 rows affected in 6 ms
oblig1> SELECT * FROM Oblig1.Film
[2022-09-11 18:19:18] 13 rows retrieved starting from 1 in 39 ms (execution: 5 ms, fetching: 34 ms)
```

som resulterer i

	FNr	Tittel	År	Land	Sjanger	Alder	Tid	Pris
1	1	Casablanca	1942	USA	Drama	15	102	149.00
2	2	Fort Apache	1948	USA	Western	15	127	<null>
3	3	Apocalypse Now	1979	USA	Action	18	155	123.00
4	4	Streets of Fire	1984	USA	Action	15	93	<null>
5	5	High Noon	1952	USA	Western	15	85	123.00
6	6	Cinema Paradiso	1988	Italia	Komedie	11	123	<null>
7	7	Asterix hos britene	1988	Frankrike	Tegnefilm	7	78	149.00
8	8	Veiviseren	1987	Norge	Action	15	96	87.00
9	9	Salmer fra kjøkkenet	2002	Norge	Komedie	7	80	149.00
10	10	Anastasia	1997	USA	Tegnefilm	7	94	123.00
11	11	La Grande bouffe	1973	Frankrike	Drama	15	129	87.00
12	12	Blues Brothers 2000	1998	USA	Komedie	11	124	135.00
13	13	Beatles: Help	1965	Storbritannia	Musikk	11	144	<null>

d)

For å finne tittel, sjanger og pris for filmer produsert i eller etter 1988, og sortere resultatet synkende etter pris, så bruker jeg følgende spørring:

```
oblig1> SELECT Tittel, Sjanger, Pris
FROM Oblig1.Film
WHERE År >= 1988
ORDER BY Pris DESC
[2022-09-11 18:13:38] 5 rows retrieved starting from 1 in 44 ms (execution: 5 ms, fetching: 39 ms)
```

som resulterer i

	Tittel	Sjanger	Pris
1	Asterix hos britene	Tegnefilm	149.00
2	Salmer fra kjøkkenet	Komedie	149.00
3	Blues Brothers 2000	Komedie	135.00
4	Anastasia	Tegnefilm	123.00
5	Cinema Paradiso	Komedie	<null>

e)

For å hente alle kolonner for filmer som mangler pris, sortert først etter aldersgrense og så alfabetisk etter sjanger, så bruker jeg følgende spørring:

```
oblig1> SELECT * FROM Oblig1.Film
WHERE Pris IS NULL
ORDER BY Alder, Sjanger ASC
[2022-09-11 18:26:48] 4 rows retrieved starting from 1 in 52 ms (execution: 18 ms, fetching: 34 ms)
```

som resulterer i

	FNr	Tittel	År	Land	Sjanger	Alder	Tid	Pris
1	6	Cinema Paradiso	1988	Italia	Komedie	11	123	<null>
2	13	Beatles: Help	1965	Storbritannia	Musikk	11	144	<null>
3	4	Streets of Fire	1984	USA	Action	15	93	<null>
4	2	Fort Apache	1948	USA	Western	15	127	<null>

f)

Jeg finner antall filmer i hver sjanger som er til salgs, og totalpris per sjanger, ved hjelp av følgende spørring:

```
oblig1> SELECT Sjanger, COUNT(Sjanger) AS 'Antall filmer', SUM(Pris) AS 'Totalpris'
FROM Oblig1.Film
WHERE Pris IS NOT NULL
GROUP BY Sjanger
ORDER BY Sjanger ASC
[2022-09-11 18:51:56] 5 rows retrieved starting from 1 in 30 ms (execution: 5 ms, fetching: 25 ms)
```

som resulterer i

	Sjanger	'Antall filmer'	Totalpris
1	Action	2	210.00
2	Drama	2	236.00
3	Komedie	2	284.00
4	Tegnefilm	2	272.00
5	Western	1	123.00

g)

Jeg legger til data for filmen "Ghost in the Shell (1995)" ved følgende spørring:

```
oblig1> INSERT INTO Oblig1.Film
VALUE (14, 'Ghost in the Shell', 1995, 'Japan', 'Tegnefilm', 13, 83, 95.00)
[2022-09-11 19:02:07] 1 row affected in 19 ms
oblig1> SELECT * FROM Oblig1.Film
WHERE Tittel = 'Ghost in the Shell'
[2022-09-11 19:04:31] 1 row retrieved starting from 1 in 29 ms (execution: 4 ms, fetching: 25 ms)
```

som resulterer i

	FNr	Tittel	År	Land	Sjanger	Alder	Tid	Pris
1	14	Ghost in the Shell	1995	Japan	Tegnefilm	13	83	95.00

h)

Braker følgende sql for å korrigere tittelen film nummer 5 fra 'High Noon' til 'High Moon':

```
oblig1> UPDATE Oblig1.Film
        SET Tittel = 'High Moon'
        WHERE Tittel = 'High Noon'
[2022-09-11 19:15:28] 1 row affected in 7 ms
oblig1> SELECT FNr, Tittel
        FROM Oblig1.Film
        WHERE Tittel = 'High Moon'
[2022-09-11 19:16:32] 1 row retrieved starting from 1 in 30 ms (execution: 5 ms, fetching: 25 ms)
```

som resulterer i

	FNr	Tittel
1	5	High Moon

i)

Øker prisen med 10% på action-filmer med følgende kode:

```
oblig1> UPDATE Oblig1.Film
        SET Pris = Pris * 1.1
        WHERE Sjanger = 'Action'
[2022-09-11 19:23:18] 3 rows affected in 6 ms
oblig1> SELECT Sjanger, Tittel, Pris
        FROM Oblig1.Film
        WHERE Sjanger = 'Action'
        ORDER BY Tittel ASC
[2022-09-11 19:27:54] 3 rows retrieved starting from 1 in 31 ms (execution: 4 ms, fetching: 27 ms)
```

som resulterer i

	Sjanger	Tittel	Pris
1	Action	Apocalypse Now	135.30
2	Action	Streets of Fire	<null>
3	Action	Veiviseren	95.70

J)

Sletter filmen 'Anastasia':

```
oblig1> DELETE
        FROM Oblig1.Film
        WHERE Tittel = 'Anastasia'
[2022-09-11 19:34:24] 1 row affected in 21 ms
oblig1> SELECT Tittel
        FROM Oblig1.Film
        ORDER BY Tittel ASC
[2022-09-11 19:35:25] 13 rows retrieved starting from 1 in 35 ms (execution: 17 ms, fetching: 18 ms)
```

og jeg ser at 'Anastasia' ikke lenger er i tabellen:

	Tittel
1	Apocalypse Now
2	Asterix hos britene
3	Beatles: Help
4	Blues Brothers 2000
5	Casablanca
6	Cinema Paradiso
7	Fort Apache
8	Ghost in the Shell
9	High Moon
10	La Grande bouffe
11	Salmer fra kjøkkenet
12	Streets of Fire
13	Veiviseren

OPPGAVE 2

a)

Lager tabellen 'Kunde':

```
oblig1> CREATE TABLE Oblig1.Kunde (
    KNr INTEGER UNSIGNED NOT NULL,
    Fornavn VARCHAR(50),
    Etternavn VARCHAR(50),
    Adresse VARCHAR(100),
    PostNr SMALLINT UNSIGNED,
    PRIMARY KEY (KNr)
)
[2022-09-11 19:45:51] completed in 37 ms
oblig1> SELECT *
FROM Oblig1.kunde
[2022-09-11 19:47:38] 0 rows retrieved in 29 ms (execution: 4 ms, fetching: 25 ms)
```

som resulterer i

KNr	Fornavn	Etternavn	Adresse	PostNr
-----	---------	-----------	---------	--------

b)

Legger til data for tre kunder:

```
oblig1> INSERT INTO Oblig1.Kunde
VALUES (10001, 'Kari', 'Mo', 'Moldegata 21 H0402', 0445),
       (10002, 'Geir', 'Gallestein', 'Grønnegata 68 H0304', 2317),
       (10003, 'Reidun', 'Roterud', 'Hylleråsvegen 11', 2440)
[2022-09-11 19:56:01] 3 rows affected in 6 ms
oblig1> SELECT *
FROM Oblig1.Kunde
[2022-09-11 19:56:03] 3 rows retrieved starting from 1 in 35 ms (execution: 5 ms, fetching: 30 ms)
```

som resulterer i

	KNr	Fornavn	Etternavn	Adresse	PostNr
1	10001	Kari	Mo	Moldegata 21 H0402	445
2	10002	Geir	Gallestein	Grønnegata 68 H0304	2317
3	10003	Reidun	Roterud	Hylleråsvegen 11	2440

c)

Lager en faktura-tabell med kolonner for fakturanummer, kundennummer, beløp, opprettelsesdato, forfallsdato, filmnummer og betalt. Fakturanummeret er unikt til hver faktura, og brukes derfor som primærnøkkel. Kundennummeret er en fremmednøkkel knyttet til hovednøkkelen i kunde-tabellen, slik at man vet hvilken kunde som har utført bestillingen. Filmnummeret er også en fremmednøkkel knyttet til hovednøkkelen i film-tabellen, slik at man vet hvilken film som er blitt leid. Til slutt har vi en bool i betalt som er True, hvis fakturaen er betalt og false hvis den ikke er betalt.

Ingen av nøklene kan være NULL, siden fakturaen må ha et fakturanummer, må kunne knyttes til en leietager, og må kunne knyttes til produktet som ble utleid, ellers er det ikke en gyldig faktura. Jeg lar beløp være NULL ved default, da må prisen føres inn manuelt. Jeg valgte å ha prisen som DECIMAL(7, 2), siden da kan beløp opp til 99.999,- brukes. Datoene er av typen DATE, slik at de er på formatet 'YYYY-MM-DD', f.eks. '2022.04.11', og de kan ikke være NULL, for da kan ikke transaksjonen tidfestes, og er ugyldig. Hvis en faktura ikke opprettes med utfylte datoer, så blir opprettelsesdatoen lik datoen når fakturaen ble lagt til i tabellen, og forfallsdato blir 14 dager fra opprettelsesdato.

```
oblig1> CREATE TABLE Oblig1.Faktura
(
    FakturaNr      INTEGER UNSIGNED NOT NULL,
    KundeNr        INTEGER UNSIGNED NOT NULL,
    FilmNr         INTEGER UNSIGNED NOT NULL,
    Beløp          DECIMAL(7, 2)      DEFAULT NULL,
    Opprettelsesdato DATE             NOT NULL DEFAULT (CURRENT_DATE),
    Forfallsdato   DATE             NOT NULL DEFAULT (Opprettelsesdato + INTERVAL 14 DAY),
    Betalt         BOOL DEFAULT (FALSE),
    PRIMARY KEY (FakturaNr),
    FOREIGN KEY (KundeNr) REFERENCES Oblig1.Kunde (KNr),
    FOREIGN KEY (FilmNr) REFERENCES Oblig1.Film (FNr)
)
[2022-09-11 22:04:56] completed in 24 ms
```

d)

Jeg legger til to fakturaer for Kari Mo:

```
oblig1> INSERT INTO Oblig1.Faktura (FakturaNr, KundeNr, FilmNr, Beløp, Opprettelsesdato, Betalt)
VALUES
(
    10001,
    (SELECT KNr FROM Oblig1.Kunde WHERE Fornavn = 'Kari' AND Etternavn = 'Mo'),
    14,
    (SELECT Pris FROM Oblig1.Film WHERE FNr = 14),
    '2021-12-30',
    TRUE
),
(
    10002,
    (SELECT KNr FROM Oblig1.Kunde WHERE Fornavn = 'Geir' AND Etternavn = 'Gallestein'),
    11,
    (SELECT Pris FROM Oblig1.Film WHERE FNr = 11),
    '2022-04-01',
    TRUE
),
(
    13592,
    (SELECT KNr FROM Oblig1.Kunde WHERE Fornavn = 'Kari' AND Etternavn = 'Mo'),
    5,
    (SELECT Pris FROM Oblig1.Film WHERE FNr = 5),
    CURRENT_DATE,
    FALSE
)
[2022-09-11 22:13:52] 3 rows affected in 7 ms
oblig1> SELECT *
FROM Oblig1.Faktura
[2022-09-11 22:13:59] 3 rows retrieved starting from 1 in 26 ms (execution: 5 ms, fetching: 21 ms)
```

som resulterer i

	FakturaNr	KundeNr	FilmNr	Beløp	Opprettelsesdato	Forfallsdato	Betalt
1	10001	10001	14	95.00	2021-12-30	2022-01-13	1
2	10002	10002	11	87.00	2022-04-01	2022-04-15	1
3	13592	10001	5	123.00	2022-09-11	2022-09-25	0

e)

Finner Kari Mo sine fakturaer ved navn:

```
oblig1> SELECT *
FROM Oblig1.Faktura
WHERE KundeNr = (SELECT KNr FROM Oblig1.Kunde WHERE Fornavn = 'Kari' AND Etternavn = 'Mo')
[2022-09-11 22:18:13] 2 rows retrieved starting from 1 in 38 ms (execution: 4 ms, fetching: 34 ms)
```

som resulterer i

	FakturaNr	KundeNr	FilmNr	Beløp	Opprettelsesdato	Forfallsdato	Betalt
1	10001	10001	14	95.00	2021-12-30	2022-01-13	1
2	13592	10001	5	123.00	2022-09-11	2022-09-25	0

f)

Lagt til forretningsregel for beløpet, slik at beløpet må være i intervallet [0, 10.000]:

```
oblig1> ALTER TABLE Oblig1.Faktura
ADD CONSTRAINT Beløpsintervall
CHECK ( Beløp >= 0.00 AND Beløp <= 10000.00)
[2022-09-11 22:28:29] 3 rows affected in 55 ms
```


g)

Prøver å legge inn faktura med ulovlig beløp:

```
oblig1> INSERT INTO Oblig1.Faktura (FakturaNr, KundeNr, FilmNr, Beløp)
        VALUE (1010, 10003, 1, 10000.69)
[2022-09-11 22:38:57] [HY000][3819] Check constraint 'Beløpsintervall' is violated.
[2022-09-11 22:38:57] [HY000][3819] Check constraint 'Beløpsintervall' is violated.
oblig1> SELECT *
        FROM Oblig1.Faktura
[2022-09-11 22:39:03] 3 rows retrieved starting from 1 in 53 ms (execution: 6 ms, fetching: 47 ms)
```

Her fikk jeg en feilmelding om at forretningsregelen er brutt, og ved å printe tabellen så ser jeg at den ulovlige fakturaen ikke ble lagt til:

	FakturaNr	KundeNr	FilmNr	Beløp	Opprettelsesdato	Forfallsdato	Betalt
1	10001	10001	14	95.00	2021-12-30	2022-01-13	1
2	10002	10002	11	87.00	2022-04-01	2022-04-15	1
3	13592	10001	5	123.00	2022-09-11	2022-09-25	0

h)

Reduserer beløpet med en faktor 10, og kjører koden på nytt. Denne gang kommer ingen feilmelding:

```
oblig1> INSERT INTO Oblig1.Faktura (FakturaNr, KundeNr, FilmNr, Beløp)
        VALUE (1010, 10003, 1, 1000.69)
[2022-09-11 22:39:01] 1 row affected in 5 ms
oblig1> SELECT *
        FROM Oblig1.Faktura
[2022-09-11 22:39:04] 4 rows retrieved starting from 1 in 29 ms (execution: 3 ms, fetching: 26 ms)
```

Jeg ser at den korrigerte fakturaen ble lagt til:

	FakturaNr	KundeNr	FilmNr	Beløp	Opprettelsesdato	Forfallsdato	Betalt
1	1010	10003	1	1000.69	2022-09-11	2022-09-25	0
2	10001	10001	14	95.00	2021-12-30	2022-01-13	1
3	10002	10002	11	87.00	2022-04-01	2022-04-15	1
4	13592	10001	5	123.00	2022-09-11	2022-09-25	0

OPPGAVE 3

a)

Dokumentasjon av SUBSTRING_INDEX():

- `SUBSTRING_INDEX(str, delim, count)`

Returns the substring from string *str* before *count* occurrences of the delimiter *delim*. If *count* is positive, everything to the left of the final delimiter (counting from the left) is returned. If *count* is negative, everything to the right of the final delimiter (counting from the right) is returned. `SUBSTRING_INDEX()` performs a case-sensitive match when searching for *delim*.

```
mysql> SELECT SUBSTRING_INDEX('www.mysql.com', '.', 2);
-> 'www.mysql'
mysql> SELECT SUBSTRING_INDEX('www.mysql.com', '.', -2);
-> 'mysql.com'
```

This function is multibyte safe.

`SUBSTRING_INDEX()` returns NULL if any of its arguments are NULL.

Hentet fra:

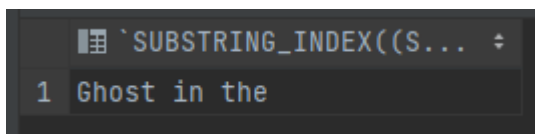
ORACLE. (2022). MySQL 8.0 Reference Manual: 12.8 String Functions and Operators [Documentation]. MySQL. <https://dev.mysql.com/doc/refman/8.0/en/string-functions.html#function-substring-index>

b)

For å vise at jeg kan bruke `SUBSTRING_INDEX()`, så bruker jeg den til å hente de to midterste ordene i tittelen til film nummer 14 (Ghost in the Shell). Først lager jeg et uttrykk som deler tittelen ved mellomrom, og returnerer en streng med teksten opp til tredje mellomrom:

```
oblig1> SELECT SUBSTRING_INDEX((SELECT Tittel FROM Oblig1.Film WHERE FNr = 14), ' ', 3)
[2022-09-11 23:43:05] 1 row retrieved starting from 1 in 21 ms (execution: 5 ms, fetching: 16 ms)
```

Den returnerer som forventet teksten 'Ghost in the':

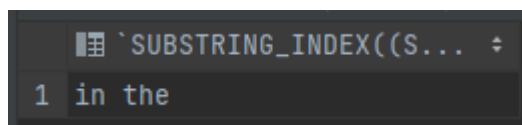


```
1 Ghost in the
```

Jeg kan så bruke det tidligere nevnte sql-uttrykket som input-streng til `SUBSTRING_INDEX()`, og med count lik `-2`, slik at jeg henter substrangen med tekst fra høyre ende av input-strengen til første mellomrom i input-strengen:

```
oblig1> SELECT SUBSTRING_INDEX((SELECT SUBSTRING_INDEX((SELECT Tittel FROM Oblig1.Film WHERE FNr = 14), ' ', 3)), ' ', -2)
[2022-09-11 23:43:05] 1 row retrieved starting from 1 in 44 ms (execution: 6 ms, fetching: 38 ms)
```

Da for jeg som forventet teksten 'in the':



```
1 in the
```

OPPGAVE 4

a)

Henter all data fra `information_schema.TABLES` med følgende spørring:

```
oblig1> SELECT *
        FROM information_schema.TABLES
[2022-09-11 23:55:12] 339 rows retrieved starting from 1 in 223 ms (execution: 52 ms, fetching: 171 ms)
```

Jeg ser så at alle tre tabellene i Oblig1 databasen har en 'InnoDB' som verdi i ENGINE-kolonnen:

TABLE_CATALOG	TABLE_SCHEMA	TABLE_NAME	TABLE_TYPE	ENGINE
def	mysql	time_zone_transition_type	BASE TABLE	InnoDB
def	mysql	user	BASE TABLE	InnoDB
def	oblig1	faktura	BASE TABLE	InnoDB
def	oblig1	film	BASE TABLE	InnoDB
def	oblig1	kunde	BASE TABLE	InnoDB
def	performance_schema	accounts	BASE TABLE	PERFORMANCE_SCHEMA

b)

Jeg fant dokumentasjonen for `information_schema` databasen her:

ORACLE. (2022). MySQL Information Schema [Documentation]. MySQL. <https://dev.mysql.com/doc/mysql-infoschema-excerpt/8.0/en/>

c)

Jeg sjekker tabellen for databasemotorer, for å se hva slags informasjon er lagret om de:

```
oblig1> SELECT *
        FROM information_schema.ENGINES
[2022-09-12 00:12:55] 9 rows retrieved starting from 1 in 29 ms (execution: 5 ms, fetching: 24 ms)
```

som gir:

ENGINE	SUPPORT	COMMENT	TRANSACTIONS	XA	SAVEPOINTS
1 MEMORY	YES	Hash based, stored in memory, useful for temporary tables	NO	NO	NO
2 MRG_MYISAM	YES	Collection of identical MyISAM tables	NO	NO	NO
3 CSV	YES	CSV storage engine	NO	NO	NO
4 FEDERATED	NO	Federated MySQL storage engine	<null>	<null>	<null>
5 PERFORMANCE_SCHEMA	YES	Performance Schema	NO	NO	NO
6 MyISAM	YES	MyISAM storage engine	NO	NO	NO
7 InnoDB	DEFAULT	Supports transactions, row-level locking, and foreign keys	YES	YES	YES
8 BLACKHOLE	YES	/dev/null storage engine (anything you write to it disappears)	NO	NO	NO
9 ARCHIVE	YES	Archive storage engine	NO	NO	NO

Jeg sjekker også tabellen for forretningsregler for å se hvordan den jeg lagde i oppgave 2. f) blir lagret:

```
oblig1> SELECT *
        FROM information_schema.CHECK_CONSTRAINTS
[2022-09-12 00:12:55] 1 row retrieved starting from 1 in 51 ms (execution: 4 ms, fetching: 47 ms)
```

som gir:

CONSTRAINT_CATALOG	CONSTRAINT_SCHEMA	CONSTRAINT_NAME	CHECK_CLAUSE
def	oblig1	Beløpsintervall	((`BelÃ,p` >= 0.00) and (`BelÃ,p` <= 10000.00))

Interessant å se at den ikke lagrer 'ø' riktig i selve betingelsen, men uten at det skaper problemer for databasens operasjon.

OPPGAVE 5**a)**

Etter å ha gjort denne obligen, så har ikke forståelsen min for konseptet bak relasjonsdatabaser endret seg særlig. Dette er fordi jeg allerede hadde lært meg mye av teorien tidligere. Derimot har min forståelse, samt beherskelse av SQL spesifikt blitt en god del bedre, og jeg har fått en god forståelse av hvordan databasene opperer i MySQL implementasjonen.

b)

Jeg ser nå hvordan jeg skal gå fram for å dele opp data i tabeller med relaterte felt, og hvordan jeg kan knytte disse tabellene sammen med bruk av fremmednøkler. Samt at jeg nå har erfaring med å designe en SQL-database, og derfor føler meg mer rustet til å designe fremtidige databaser.