Køreplan for databaser 2. gang

```
/*
Opgave 1

    Det er databasen Employees der
    skal bruges til hele opgaven
*/
use employees;
/*
1.1
        Du skal hente alle kolonner og alle data
    fra tabellen employees

        Resultat: 300.024 rows
*/
select * from employees;

/*
1.2
        Du skal hente alle data
    fra tabellen employees
    men kun first_name, last_name og gender

        Resultat: 300.024 rows
*/
select
        first_name,
        last_name,
    gender
from employees;

/*
1.3
        Du skal hente de forskellige
    titler fra tabellen titles
    Hver titel skal kun vises en gang

    Resultat: 7 Rows
*/
select distinct title
from titles;
```

```
/*
1.4
        Du skal hente alle ansatte fra
    tabellen employees der er kvinder.
    Sorter i faldende orden efter first_name

    Resultat: 120.051 Rows
*/
select * from employees
where gender = 'F'
order by first_name DESC;

/*
1.5
        Du skal hente alle ansatte fra
    tabellen employees der er kvinder

    Resultat: 120.051 Rows
*/
select * from employees
where gender = 'F';



/*
1.6
        Find alle ansatte fra tabellen employees
    der har fødselsdag i december måned

    Tip: brug MONTH(birth_date)

    Resultat: 25.326 Rows
*/
select * from employees
where MONTH(birth_date) = 12;

/*
1.7
        Find alle ansatte fra tabellen employees
    der er ansat i enten november eller december

    Tip: brug MONTH(hire_date)

    Resultat: 49.826 Rows
*/
select * from employees
where month(hire_date) = 11
```

```
or month(hire_date) = 12;

/*
1.8
        Find alle ansatte fra tabellen employees der
    er ansat i november måned 1991

    Tip: brug YEAR(hire_date)

    Resultat: 1.779 Rows
*/
select * from employees
where year(hire_date) = 1991
and month(hire_date) = 11;

/*
1.9
        Find alle ansatte fra tabellen employees der
    IKKE er ansat i november måned 1991
    Sorter efter hire_date i stigende orden

    Resultat: 255.373 Rows
*/
select * from employees
where not year(hire_date) = 1991
and not month(hire_date) = 11
order by hire_date;

/*
1.10
        Find alle ansatte fra tabellen employees
    der har et emp_no mellem 20000 og 20500

    Resultat: 501 Rows
*/
select * from employees
where emp_no between 20000 and 20500;

/*
1.11
        Find alle ansatte fra tabellen employees
    der er ansat i følgende måneder:

    Januar - 1
    Marts - 3
    Maj - 5
```

```
   November - 11

   Sorter efter hire_date i stigende orden

   Resultat: 100.581 Rows
*/
select * from employees
where month(hire_date) in (1, 3, 5, 11)
order by hire_date;
```

Nyt materiale 2. gang

Datatyper

```
Create database test;

create table test
(
va varchar(1)
)

use textchar;

insert into test (va)
values('Christian');

insert into test (va)
values('C');

create table test2
(
va varchar(255)
);

insert into test2(va)
values('12345678901234567890112345678901234567890112345678901234
56789011234567890123456789011234567890123456789011234567890123
45678901123456789012345678901123456789012345678901123456789012
34567890112345678901234567890112345678901234567890112345678901
23456789011234567890123456789011234567890123456789011234567890
12345678901123456789012345678901123456789012345678901123456789
```

012345678901');

INT unsigned

```
Create table if not exists test3
(
Number1 INT unsigned
);

insert into test3(number1)
values(4294967295);
```

FLOAT VS DECIMAL

```
create table numbers(
id INT primary key not null auto_increment,
da decimal(10,2),
db decimal(10,2),
fa float,
fb float

insert into numbers(da, db, fa, fb)
values(0.10, 0.20, 0.1, 0.2);

select da + db, fa + fb from numbers;

select da + db = 0.3 from numbers; == true or 1

select fa + fb = 0.3 from numbers; == false or 0

);
```

ENUM og SET

Her kan der vælges flere værdier fra set listen.

```
CREATE TABLE setTest(
  attrib SET('bold','italic','underline')
);
INSERT INTO setTest (attrib) VALUES ('bold');
INSERT INTO setTest (attrib) VALUES ('bold,italic');
INSERT INTO setTest (attrib) VALUES ('bold,italic,underline');
```

Her kan der kun vælges en enkelt værdi fra enumlisten.

```
CREATE TABLE enumTest(
 color ENUM('red','green','blue')
);

INSERT INTO enumTest (color) VALUES ('red');
INSERT INTO enumTest (color) VALUES ('gray');
INSERT INTO enumTest (color) VALUES ('red,green');
```

TIME ZONE

```
 select now();

  SET time_zone = '+07:00';

 show variables like '%time_zone%';

 */

 /*

 Nyere versioner af SQL understøtter ikke timestamp, fordi den udgår i 2038.

 create database dateandtime;

 use dateandtime;

 create table temp(
id INT unsigned unique auto_increment primary key,
stamp timestamp,
name varchar(64)
);

insert into temp (name)
values
('this'),
('that'),
('other');

Select * from temp;
```

Intet resultat i timestamp rækkerne

```
use dateandtime;

SET time_zone = '+02:00';

create table if not exists tempdatetime(
id INT unsigned unique auto_increment primary key,
stamp datetime default current_timestamp on update current_timestamp,
(ÆNDRE DENNE LINJE KUN)
name varchar(64)
);


insert into tempdatetime (name)
values
('this'),
('that'),
('other');

Select * from tempdatetime;

Resultat I timestamp rækkerne

*/




-- Database
CREATE DATABASE Kunder;
-- DROP DATABASE Kunder;
USE Kunder;

-- Tabel
CREATE TABLE Kunder
	(
		Kunde_Id INT PRIMARY KEY,
		Kunde_Navn VARCHAR(60)
	);

-- DROP TABLE Kunder;

CREATE TABLE Kunder
	(
		Kunde_Id INT,
		Kunde_Navn VARCHAR(60),
```

```
        PRIMARY KEY (Kunde_Id)
        AUTO_INCREMENT(kunde_Id)
    );
```

Eller

```
CREATE TABLE Kunder
(
kunde_id INT PRIMARY KEY,
kunde_navn VARCHAR(60)
);
```

Hvad betyder det så at have en primary key?

Første gang går det fint.

```
INSERT INTO kunder(kunde_id, kunde_navn)
VALUES(1, "john")
```

Anden gang går det KNAP så fint (error duplicate entry):

```
INSERT INTO kunder(kunde_id, kunde_navn)
VALUES(1, "john")

-- DROP TABLE kunder;
```

Det er nemmere at skrive det ud i en køre, hvis man ønsker en primary key som auto_incrementer

```
CREATE TABLE kunder
    (
        kunde_id INT PRIMARY KEY AUTO_INCREMENT ,
        kunde_navn VARCHAR(60)
    );
```

Vi behøver ikke at skrive kunde_id på når vi indsætter, fordi vi har sat den til at auto_incremente.

```
insert into kunder(kunde_navn)
values("john")

insert into kunder(kunde_navn)
values("john")

insert into kunder(kunde_navn)
```

values("john")

Der kan også være to primary keys, hvis man fx. Har nogle produkter i forskellige farver.

```
create database if not exists produkter;

use produkter;

CREATE TABLE if not exists produkter
	(
		produkt_type VARCHAR(20),
		produkt_farve VARCHAR(20),
		CONSTRAINT PRIMARY KEY(produkt_type, produkt_farve)
	);


insert into produkter(produkt_type, produkt_farve)
values('produkt1', 'grøn');

insert into produkter(produkt_type, produkt_farve)
values('produkt1', 'grøn');

insert into produkter(produkt_type, produkt_farve)
values('produkt1', 'blå');
```

-- ALTER

```
Use kunder;

ALTER TABLE Kunder
ADD Kunde_Adresse VARCHAR(50);

ALTER TABLE Kunder
MODIFY COLUMN Kunde_Adresse VARCHAR(80);

ALTER TABLE Kunder
DROP COLUMN Kunde_Adresse;

-- PK
ALTER TABLE Kunder
ADD PRIMARY KEY (Kunde_Id);

-- PK DROP
ALTER TABLE Kunder
```

```
DROP PRIMARY KEY;
```

-- NULL

```
use northwind;

select *
from Customers
where FAX is null;
```

Hvis hvordan feltet NOT NULL påvirker hvordan insert bruges, skift feltet.

```
DROP TABLE Kunder;

	CREATE TABLE	Kunder
	(
		Kunde_Id INT PRIMARY KEY AUTO_INCREMENT,
		Kunde_Navn VARCHAR(60) NOT NULL,
		Kunde_Adresse VARCHAR (50)
	);

INSERT INTO Kunder (kunde_adresse)
VALUES (
	'Olsen'
);

SELECT * FROM Kunder;
```

Den vil simpelthen bare have noget i feltet NOT NULL.

```
DROP TABLE Kunder;

	CREATE TABLE	Kunder
	(
		Kunde_Id INT PRIMARY KEY AUTO_INCREMENT,
		Kunde_Navn VARCHAR(60),
		Kunde_Adresse VARCHAR (50) NOT NULL
	);

INSERT INTO Kunder (kunde_adresse)
VALUES (
	'Olsen'
```

```
);

SELECT * FROM Kunder;


-- DEFAULT


CREATE TABLE Kunder
	(
		Kunde_Id INT PRIMARY KEY,
		Kunde_Navn VARCHAR(60),
	Kunde_OpretDate DATETIME DEFAULT NOW()
	);

INSERT INTO Kunder (Kunde_id, Kunde_Navn)
VALUES (
	2,
   'Hansen'
);

select * from Kunder;

ALTER TABLE Kunder
ALTER COLUMN Kunde_OpretDate DROP DEFAULT;

-- UPDATE

use kunder;

drop table kunder;

CREATE TABLE Kunder
	(
		Kunde_Id INT,
		Kunde_Navn VARCHAR(60),
		Kunde_Efternavn VARCHAR(60),
		PRIMARY KEY (Kunde_Id)
	);

INSERT INTO Kunder
VALUES(
	1,
   'Hansen',
   'Flemming'
),
```

```
(
        2,
    'Olsen',
    'Bente'
);

SELECT * FROM Kunder;

UPDATE Kunder
SET Kunde_Navn = 'Hellstern', Kunde_Efternavn = 'Hansen'
WHERE Kunde_Id = 1;
```