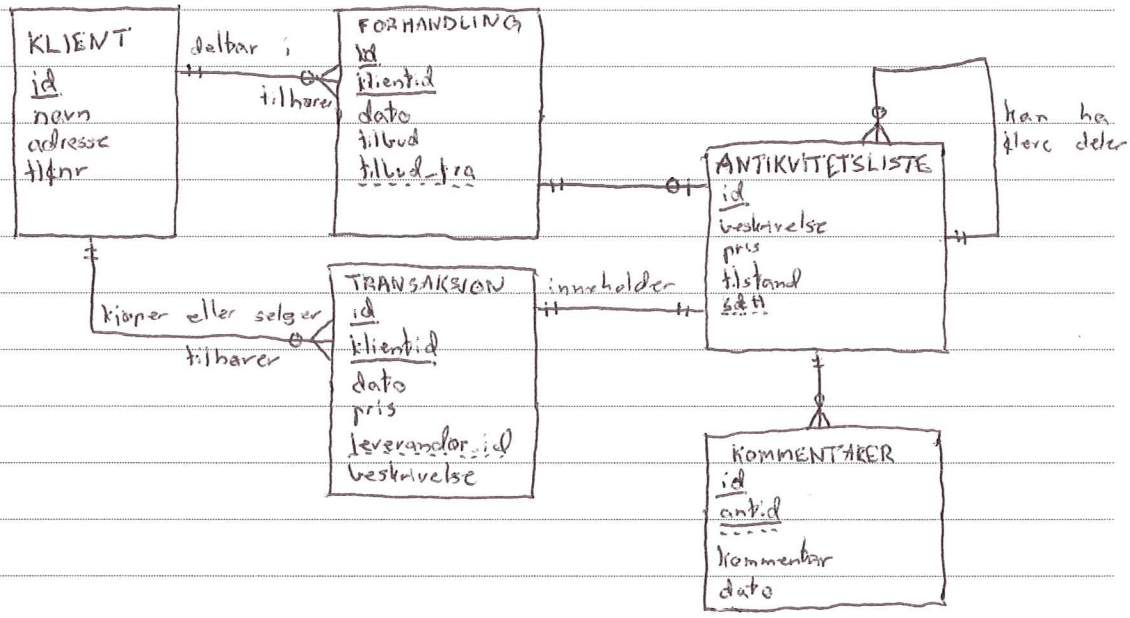
Denne kolonne er forbeholdt sensor.

Oppgave 1 a)
Databaser eliminerer redundans i stor grad hvis referanse og entitet integriteten opprettholdes. Det vil si at for hver instanse i relasjonen eksisterer det et attributt som identifiserer instansen unikt. Relasjonen unngår også redundans ved at alle relasjonene er i tredje normalform (3NF). Om databasen har redundans eller ikke er altså i stor grad bestemt av hvor godt databasen er analysert og designet.

Oppgave 1 b)
Jeg er uenig i utsagnet. Si at det eksisterer en relasjon med attributtene A, B og C der A er identifikator. Relasjonen er i 2NF kun hvis B og C er fullstendig funksjonelt avhengig av A og bare A(A determinerer verdiene til B og C). Det er altså ingen selfølge at relasjonen er i 3NF ved å ha atomisk nøkkel. Det kan for eksempel hende at verdien C er funksjonelt avhengig av både A og B. Og hvis B determinerer C har relasjonen en transitiv funksjonell avhengighet som må fjernes for å oppnå 3NF.

Denne kolonne er forbeholdt sensor.

Oppgave 2



Denne kolonne er forbeholdt sensor.

Oppgave 3

1NF: Monthreport → date, wardno, doctor-in-charge, doctor-phone, nurse, nursephone, nurseadress, patientno, lastname, firstname, homecity, visitdate, diagnosis, referred_to

Nurseadress er en multivalued attributt men velger å beholde den vendret siden den inneholder husnr, gateadresse og bynavn. Hvis jeg skulle hatt en egen relasjon til attributten ville det bare gått til tre nye tabeller (ga husnr, gateadresse og by). Dette ser jeg som unødvendig bruk av lagringskapasitet så jeg velger å beholde den slik den er. (Denormalisere)

2NF: Monthreport (wardno, date)
Nurse (name, phone, adress, wardno)
Patient (patientno, lastname, firstname, homecity)
Diagnosis (nursename, patientno, visitdate, description, referred_to)
Ward (wardno, doctor-in-charge, doctorphone)

3NF: Ward (wardno, doctor-in-charge, doctorphone)

transitiv avhengighet

Ward (wardno, doctor-in-charge)
Doctor (doctorname, doctorphone)

Denne kolonne er forbeholdt sensor.

Oppgave 4 a)

Constraint er en begrensning i tabellen som bestemmer og holder integriteten i dataene på plass. De ulike typene er: "Not null", "Primary Key" "Foreign Key", de forskjellige datatypene (varchar, integer numerie osv) for hver kolonne er også en type constraint.

```
alter table ordredetalj (
    add constraint od_pk Primary key (ordrenr, produktnr)
    add constraint od_fk_ordre Foreign key ordrenr
        REFERENCES ordre(ordrenr)
    add constraint od_fk_produkt Foreign key produktnr
        REFERENCES produkt(produktnr)
);
```

Oppgave 4 b)

Et view er en virtuell tabell som brukes av brukere av databasen. Fordelen med views er at en kan bestemme utsnitt av en eller flere tabeller med ønskelig tilgangsrettigheter. Med views kan en altså bestemme hvilke kolonner (evt rader) som skal vises eller ikke vises for brukerene.

```
create view test as (
    select kundenavn, gate, stedsnavn
    from (kunde inner join poststed USING postnr)
    order by postnr
);
```

Denne kolonne er forbeholdt sensor.

Oppgave 4 c)

```
select kundenr, kunde_navn
from kunde
where kundenr NOT IN
    (select kundenr from ordre)
```

Oppgave 4 d)

```
select k1.kundenavn, k2.kundenavn
from (kunde AS k1 inner join kunde AS k2 USING
    kunde_nr_eier)
```

Oppgave 4 e)

```
select kundenr, kundenavn, sum(antall_bestilt) Bestilt,
    sum(antall_bestilt * enhetspris)
from (kunde inner join ordre USING kundenr)
inner join ordredetalj on ordre USING ordrenr
inner join produkt on ordredetalj USING produktnr
group by kundenr, kundenavn
```

Oppgave 4 f)

Lik den over men setter inn where betingelse mellom den siste inner join og group by

```
inner join ...
where postnr between 4600, 4699 AND Bestilt > 5
group by...
```