Open Data School

virkdata

Published
with GitBook

Table of Contents

Virk Data Open Data School

Denne bog har til formål at give en introduktion til brugen af åbne data ved hjælp af en række mindre øvelser.

Åbne data kommer i mange former og størrelser, og vi vil med denne bog gerne give en introduktion til, hvordan man kan bruge nogle af de mange typer af åbne data. Hvis du har input til bogen eller vil udvikle videre på den er du meget velkommen.

Bogen er opbygget som en række øvelser, som gradvist bliver sværere samtidig med, at den inddrager forskellige datakilder. Endemålet er, at du som læser har fået en grundlæggende indsigt i, hvordan offentlige åbne data kan omsættes til indsigt og viden. Vi håber du vil kunne bruge dette afsæt til at finde på egne ideer og måder at omsætte åbne data til værdi.

Denne bog er udviklet som open source, og vi vil opfordre alle til at bruge den i lige den kontekst, der giver mening for dem. Ligeledes vil vi også opfordre til at komme med tilføjelser, enten direkte som pull requests, læs mere her eller ved at kontakte os.

Åbne data

Der er allerede fri adgang til en lang række åbne data og åbnes løbende op for tilgængeligheden til åbne data. Disse data kan blandt andet fremsøges på Virk Data.

Andre kilder til data

Kortforsyningen Udstiller danske geografiske data

Opendata.dk Udstiller en række kommuners åbne data.

Danmarks Miljøportal udstiller en række data på miljøområdet.

OpenstreetMap

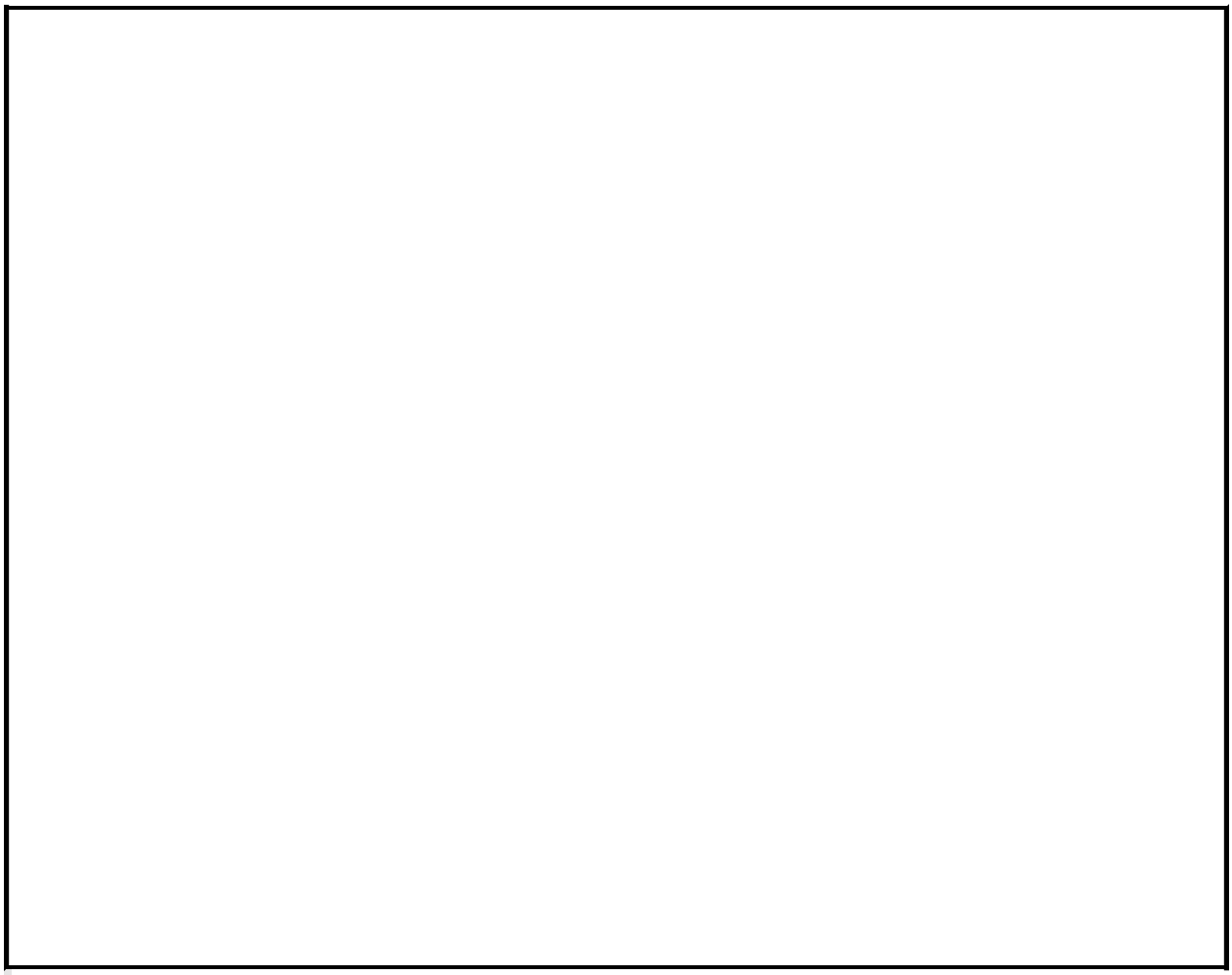
Momsdata

Data fra SKAT om virksomheders momsomsætning kan findes på Virk data

> I videoen nedenfor vises, hvordan det gøres.

Man kan følge disse trin:

1. Åben din internet browser og navigér til http://data.virk.dk/
2. Klik på fanen datakatalog. I søgefeltet skrives "moms" og der tastes ENTER
3. I søgeresultaterne klikkes på "momsomsætning (gennemsnitlig)
4. I overskriften "Dataressourcer" klikkes på resultatet "2012 Gennemsnitlig moms" og der klikkes "pil ned" og vælges "download"



> Siden med data kan findes med direkte link her: http://datahub.virk.dk/dataset/momsomsaetning-gennemsnit - Direkte download til datasættet

Danmarks administrative geografiske inddeling

Geodatastyrelsen publicerer løbende datasæt om Danmarks geografiske inddeling (DAGI) og det kan hentes på Kortforsyningen

Datasættet består af:

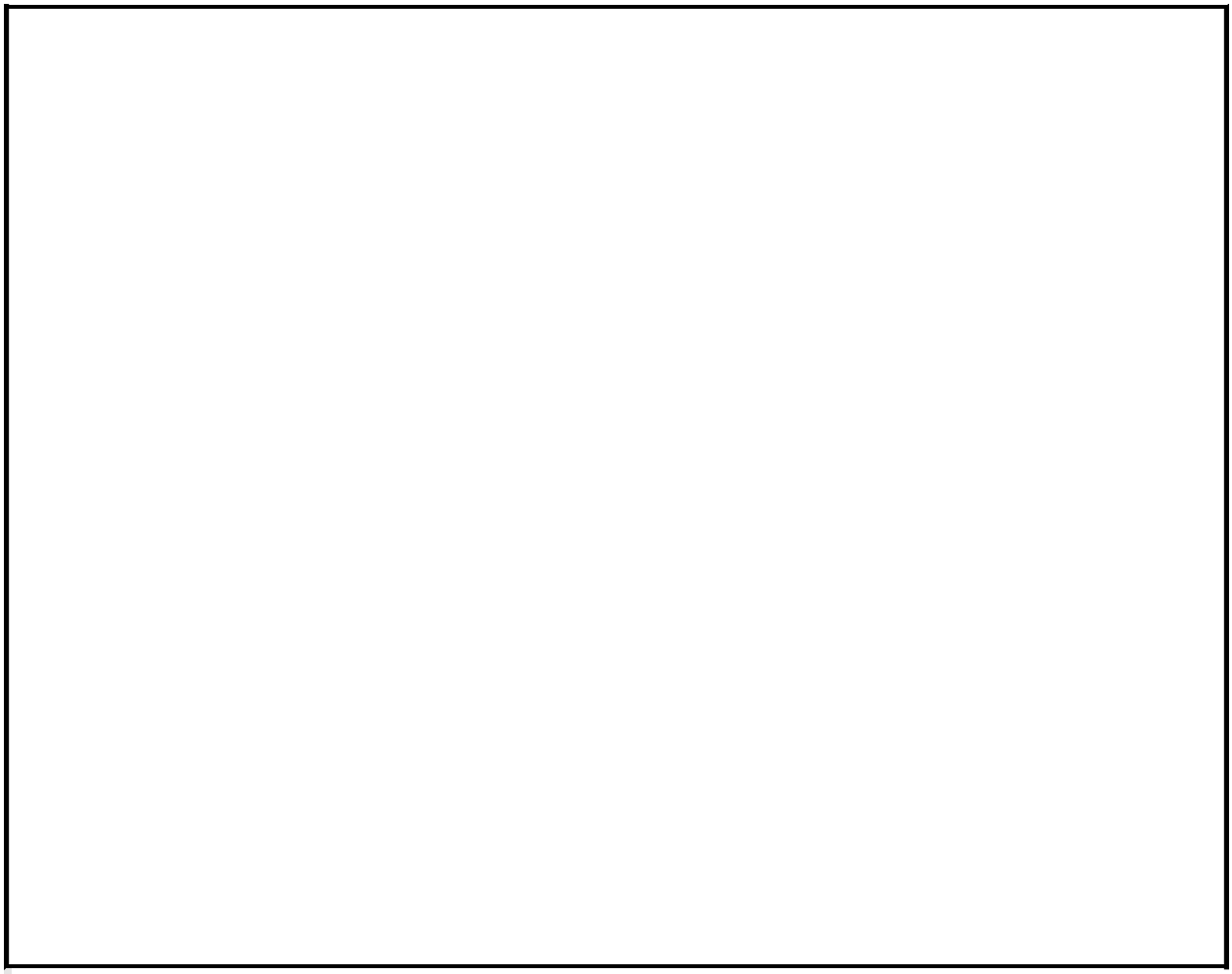
- Kommuneinddelinger
- Politikredse
- Sogne
- Postnummerinddelinger
- Regionsinddelinger
- Retskredse

Disse geografiske inddelinger er grundlaget for at danne en lang række visualiseringer på kort.

> I videoen nedenfor vises, hvordan det gøres.

Man kan følge disse trin:

1. Opret dig som bruger på Kortforsyningen
2. Log ind og under "korttype" vælges "landinddelinger"
3. Klik på Danmarks Administrative Geografiske Inddeling 1:500.000
4. Klik næste og vælg Format: SHP og Koordinatsystem: UTM/ETRS89 og klik næste
5. Klik "Læg i kurv" og klik "gennemfør" bestilling
6. Derefter er der et link til at downloade ordrer



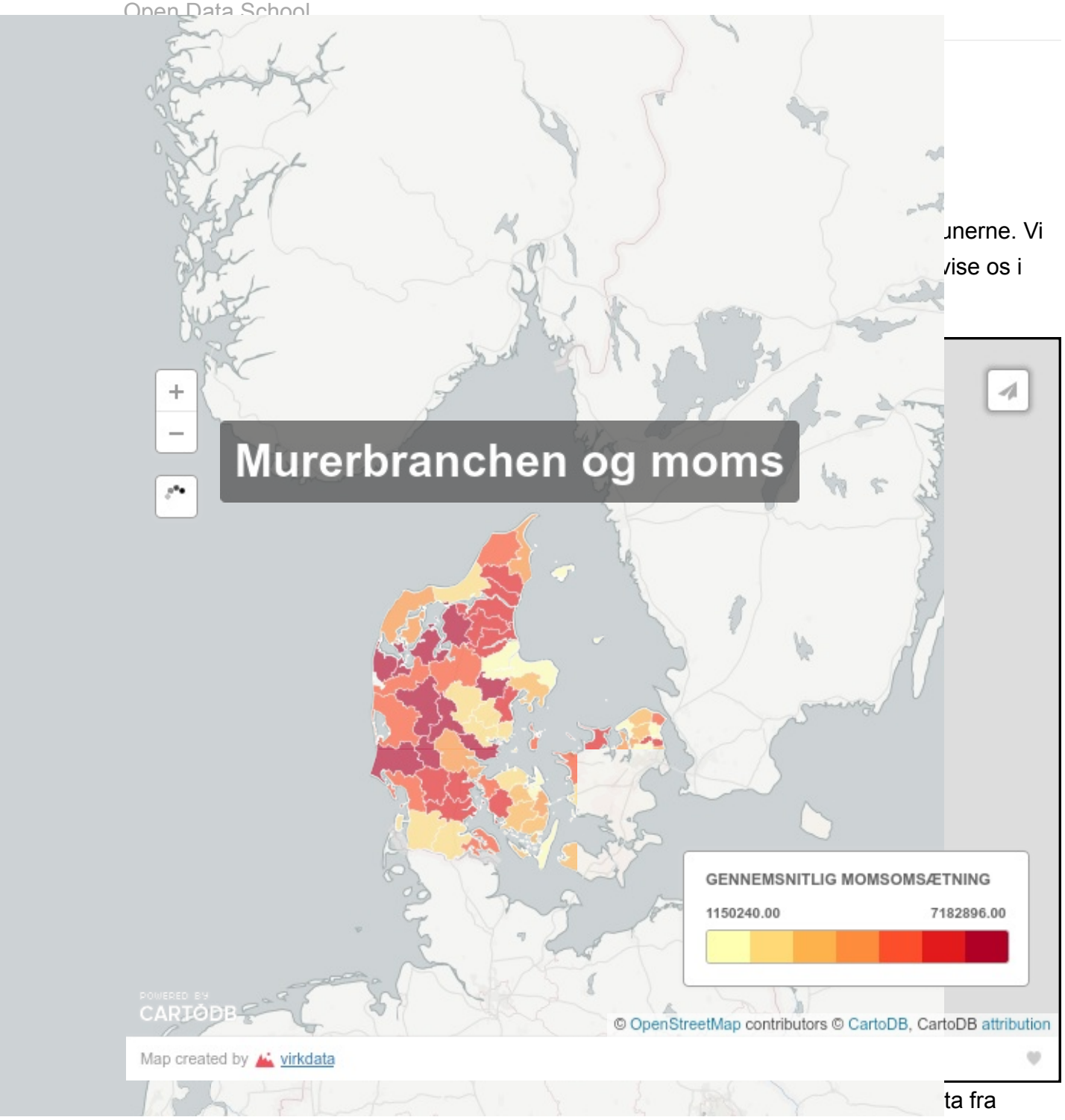
> Der vælges her et datasæt, hvor der er foretaget en udtynding i antallet af koordinater. Dette gør datasættet mindre og egner sig fint til datavisualisering af hele Danmark samlet. Hvis der ønskes et referencedatasæt vælges istedet Danmarks Administrative Geografiske Inddeling 1:10.000

Danmarks Centrale Virksomhedsregister (CVR)

Data fra CVR er en del af de åbne data. Man kan oprette sig som bruger og hente et udtræk af data her.

Til denne øvelse er der forberedt et statisk datasæt, som man kan downloade Her

unerne. Vi
vise os i



ta fra Geodatastyrelsen til at producere en datavisualisering på et kort. Kortet skal vise momsomsætning fordelt på branche og kommune. Til visualisering af data på kort anvendes CartoDB, som er et open source projekt. CartoDB som er den hostede cloud version, kan anvendes gratis med op til 50m mb data og 10 tabeller.

--	--
Datasæt	Momsdata, kommunegrænser
Sværhedsgrad	Middel
Forudsætninger	Kendskab til SQL

Data til momskortet

Til denne case anvendes dels data fra SKAT for momsomsætning fordelt på virksomheder og dels de kommunale administrative grænser fra Geodatastyrelsen. Se afsnit om data

CartoDB

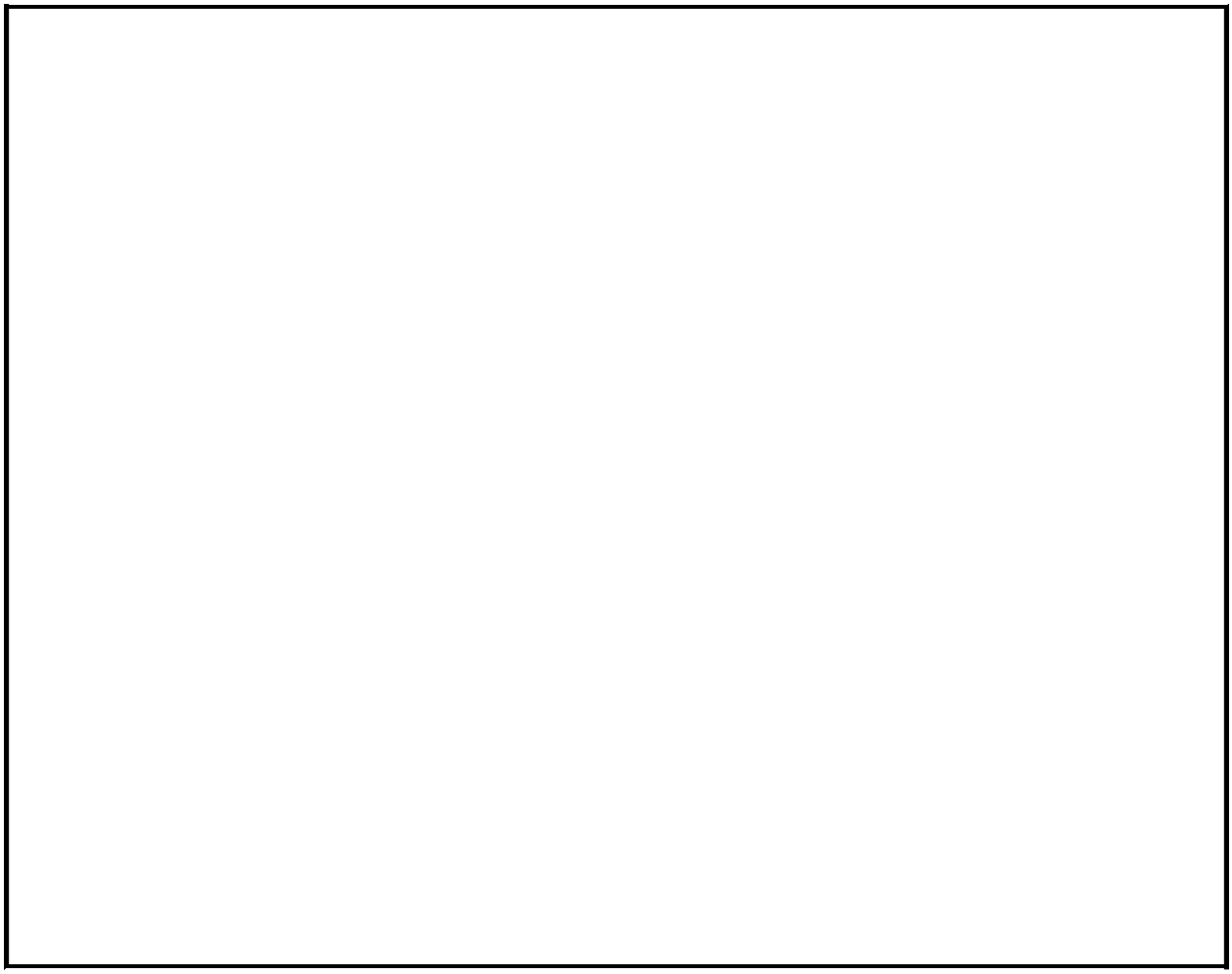
I denne CASE bruger vi CartoDB som er baseret på en PostgreSQL database med funktionalitet til geografiske data og analyser med udvidelsen PostGIS. Når data er indlæst i CartoDB, kan vi dermed bruge SQL til at udforske vores datasæt samt manipulere vores data i databasen. CartoDB er udstyret med en SQL editor direkte i webbrowseren så vi let kan udforske og klargøre vores indlæste data. Lær mere om SQL. Lær mere om geografiske analyser med PostGIS i CartoDB http://docs.cartodb.com/tips-and-tricks.html#geospatial-analysis

I denne CASE udnytter vi SQL funktioner fra både PostgreSQL og PostGIS til dels at klargøre vores indlæste data og dels til at JOINE vores to datasæt.

> I videoerne nedenfor vises de samme trin, mens de udføres i CartoDB

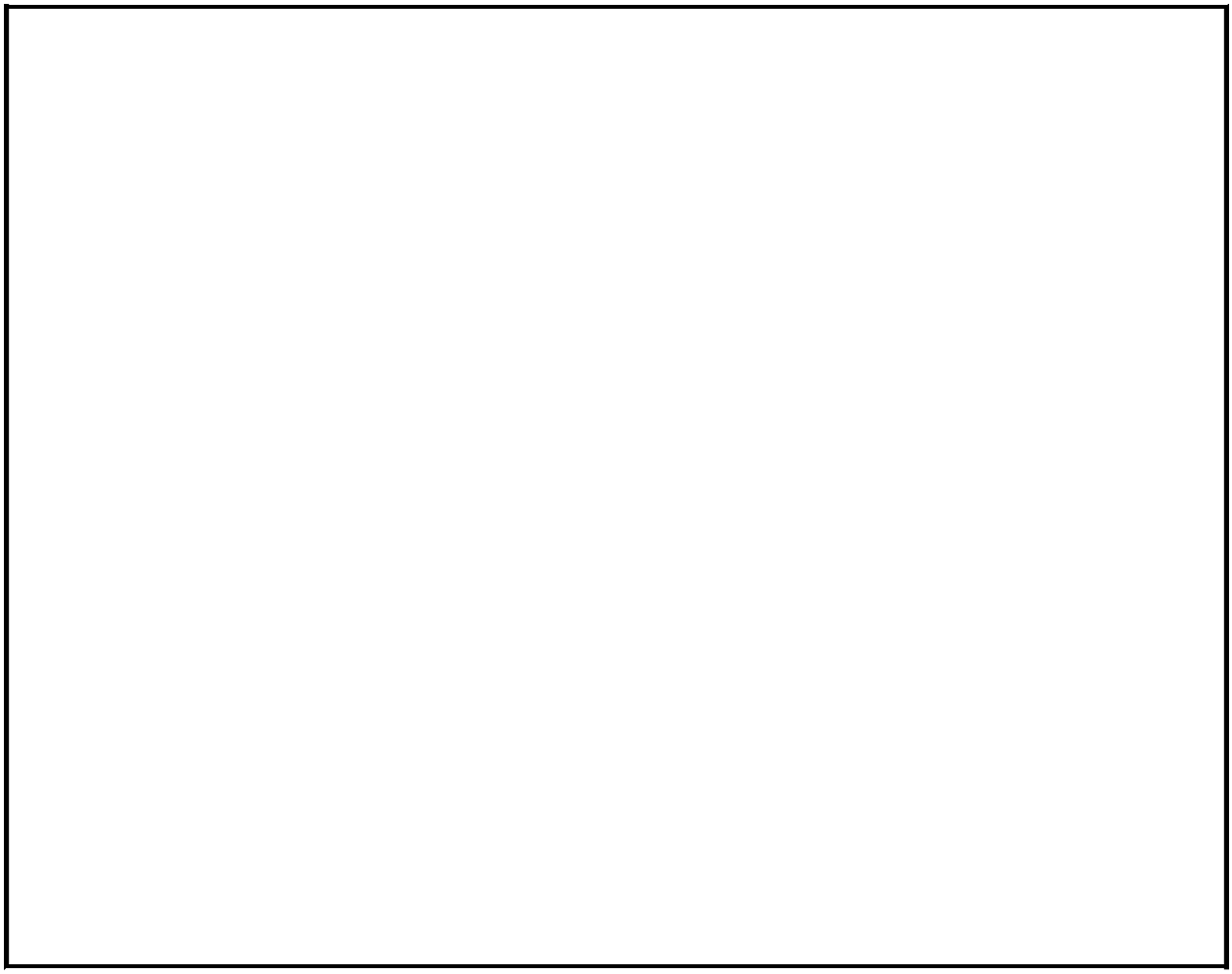
Upload moms-data til CartoDB

1. Opret en konto hos CartoDB
2. Klik "view your tables"
3. Klik "New table" og vælg filen fra Moms-data
4. Når filen er uploaded klikkes på kolonne navnene og de omdøbes så vi har en kolonne for: komnavn,branche og omsaetning



Upload kommunegrænser til CartoDB

1. Klik "view your tables"
2. Udpak filen på din computer og vælg alle filerne med navnet KOMMUNE.* og pak dem i en ny .zip fil
3. Klik "New table" i cartodb og vælg filen fra DAGI-data
4. Upload denne nye zip fil med kommunegrænser til CartoDB



Klargøring af data

Førend de to datasæt kan forenes (joines) korrekt, er det nødvendigt, at de deler en fælles nøgle. I denne case skal data joines på kommune. DAGI-datasættet med kommunegrænser indeholder en kolonne for kommunekoden og en kolonne for kommunenavnet, mens moms-datasættet alene indeholder en kolonne for kommunenavn. Derfor er vi nødsaget til at bruge kolonnen for kommunenavn til vores data-join. Derudover bruges ikke nøjagtigt samme kommunenavne i de to datasæt. Vi vil derfor forsøge at manipulere data således, at der er to kolonner med samme stavemåde på kommunenavne. Desuden har vi en udfordring med antallet af kommunepolygoner i DAGI-datasættet. Lad os starte med den.

> Det er ikke optimalt at joine på et tekstfelt. En kolonne med kommunekode bestående af heltal vil være meget bedre at anvende til at joine de to datasæt.

Kommunegrænser

Hvis vi hurtigt inspicerer vores datasæt for kommunerne, kan vi se, at der er langt flere rækkker end der er kommuner i Danmark. Det skyldes måden datasættet er konstrueret på. Geometritypen i datasættet er defineret som en POLYGON og ikke en MULTIPOLYGON. Det bevirker at polygoner, der ikke hører sammen geometrisk har en selvstændig række i datasættet. Til en kommune kan der oftes knyttes flere enkelt polygoner. F.eks. har Amager en selvstændig polygon, som ikke hører til Indre København. Det giver problemer, når vi skal joine til moms-datasættet. Vi kan med SQL lægge kommunepolygonerne sammen for de rækker, der hører til samme kommune.

Hvis vi tæller antallet af rækker, kan vi hurtigt se, at der er flere (311) rækker end de 98 kommuner. Tryk på SQL i højre side, i tabellen i cartodb, og skriv:

```
SELECT COUNT(*) FROM kommune
```

> COUNT er en AGGREGATE funktion i SQL, der bla. kan bruges til at lave simple statistikker på datasættet. Vi anvender SQL SELECT som kun udvælger i data. Vi retter IKKE i datasættet med SELECT.

Vi bruger en særlig POSTGIS funktion ST_UNION til at lægge polygoner for samme kommune sammen.

```
SELECT st_union(the_geom) as the_geom, komnavn FROM kommune GROUP by komnavn
```

Klik på linket "create table from query" og navngiv den nye tabel kommune_union

> Når man anvender GROUP BY i SQL, lægger man rækker sammen. Derfor skal øvrige kolonner aggregeres. Her lægges geometrierne sammen med en aggregate funktion, som hedder ST_Union() for alle de rækker med samme kommunenavn.

Kommunenavne

Da CartoDB er baseret på SQL, vil vi manipulere data med SQL statements. Vi vil oprette en ny kolonne i vores datasæt for kommunegrænser som stemmer overens med dén navngivning af kommunerne, der findes i moms-datasættet.

1. Klik på SQL fanen i tabelvisningen af kommunegrænser
2. Opret ny kolonne med følgende SQL

```
ALTER TABLE kommune_union ADD column komnavn_moms character varying
```

Opdatér kolonnen med de eksisterende kommunenavne og endelsen "Kommune". Vi starter med at tilføje denne endelse i vores nye kolonne.

```
UPDATE kommune_union SET komnavn_moms = komnavn || ' Kommune'
```

Vi kan nu lave et JOIN for at set om alle vores rækker i vores redigerede kommunetabel kan JOINES på momsdatasættet

Vi bruger LEFT JOIN så vi sikrer os, at vi får alle rækker fra kommunegrænserne og kan se, hvor det ikke har været muligt at joine de to datasæt

```
SELECT kom.the_geom,
kom.komnavn_moms,
skat.komnavn,
skat.omsaetning,
skat.branche
FROM kommune_union kom
LEFT JOIN skat_momsomsaetning_2012 skat ON (kom.komnavn_moms=skat.komnavn
  AND skat.branche='Murere')
  ORDER BY kom.komnavn_moms
```

Vi kan se, at der er en række kolonner fra moms-tabellen (komnavn,omsaetning,brancnhe), der har null-værdier. Det skyldes, at databasen ikke har kunne JOINE felterne, og at de altså ikke er ens (endnu). Det første, der springer i øjnene er, at Århus staves med Aa i DAGI-datasættet, mens det staves med Å i moms-datasættet. Lad os rette vores kommune-tabel de steder, hvor stavemåden stadig er forskellig.

```
UPDATE kommune_union SET komnavn_moms = 'Århus Kommune' WHERE komnavn_moms = 'Aarhus Komm
UPDATE kommune_union SET komnavn_moms = 'Brønderslev-Dronninglund Kommune' WHERE komnavn_
UPDATE kommune_union SET komnavn_moms = 'Høje-Taastrup Kommune' WHERE komnavn_moms = 'Høj
UPDATE kommune_union SET komnavn_moms = 'Københavns Kommune' WHERE komnavn_moms = 'Københ
```

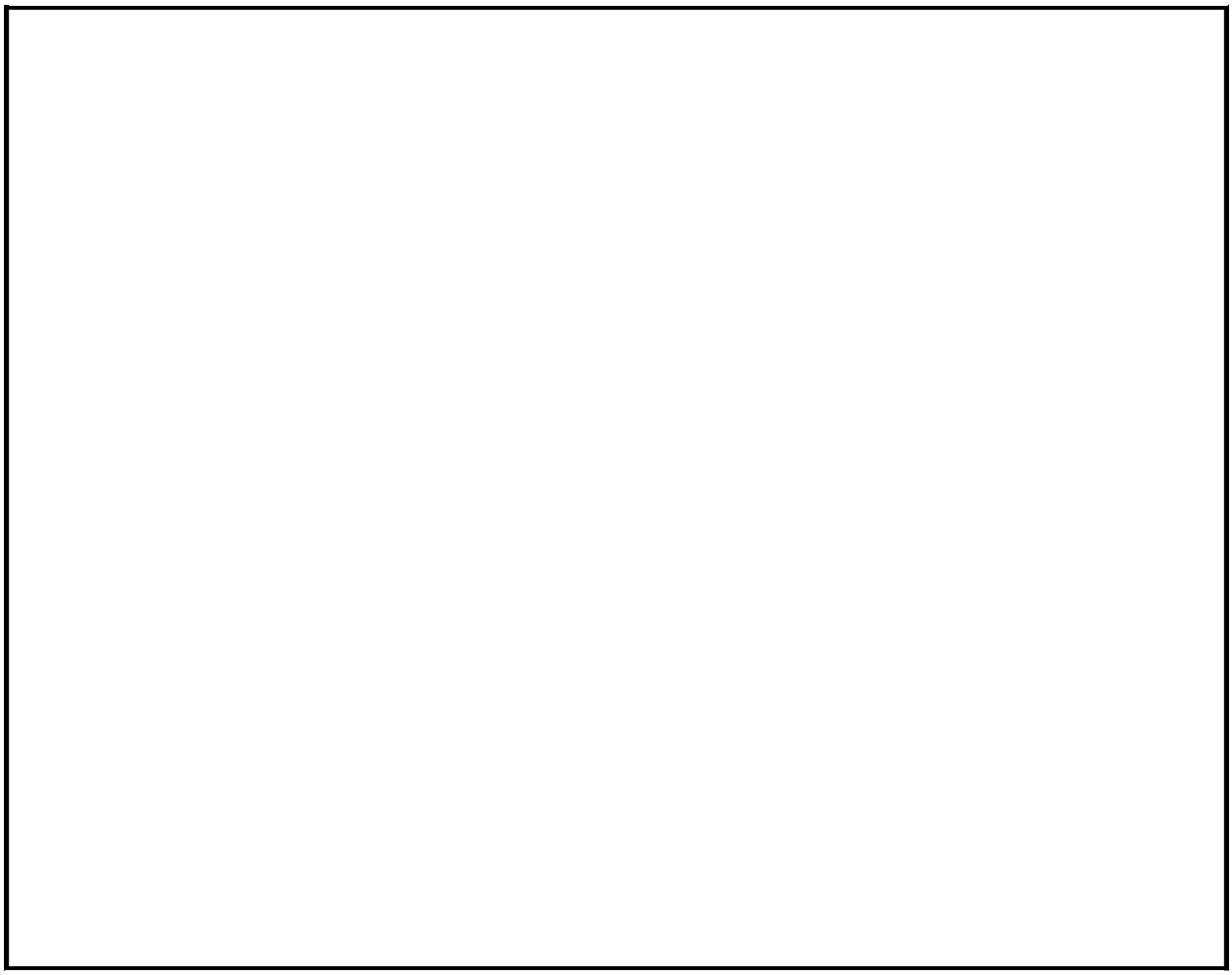
Når vi afvikler ovenstående JOIN kan vi se, at der stadig er rækker, der ikke JOINEs fordi ordet "Kommune" i nogle rækker er stavet med lille "k". Vi opdaterer moms-datasætte og sikrer, at Kommune er stavet med stort "K"

```
UPDATE skat_momsomsaetning_2012 set komnavn = replace(komnavn, 'kommune', 'Kommune')
```

Der er stadig udfordringer. Christiansø er ikke en kommunal administrativ inddeling og findes ikke i moms-datasættet. Vi vælger at slette denne i vores kommunegrænser.

```
DELETE FROM kommune_union WHERE komnavn_moms = 'Christiansø Kommune'
```

Endelig kan vi se, at den eneste kommune, der ikke kan joines, er Norddjurs Kommune. Den findes ikke i moms-datasættet. Det kan muligvis skyldes, at der er mindre end fem virksomheder af den type i kommunen. Disse medtages ikke kan man se på Virk Data



Vi er nu klar til at producere selve kortet

Kort med CartoDB

I Cartodb skelnes der mellem "tables" og "visualizations"

En tabel repræsenterer de rå data i Cartodb-databasen

Vi har nu forberedt to tabeller:

1. En tabel med kommuneafgrænsninger med kommunenavne svarende til de navne, der findes i momsdatasættet. Vi har derudover lagt polygoner sammen, således at der kun findes én mulitipolygon pr. kommune
2. En tabel med den gennemsnitlige momsbetaling fordelt på kommune og branche

En visualisation repræsenterer en særlig visualisering af en eller flere tabeller

I dette tilfælde vil vi joine de to tabeller on-the fly for at vise den gennemsnitlige momsbetaling i murerbranchen fordelt på kommuner.

Visualisation

For at vi kke bruger for meget plads på vores CartoDB konto vil vi ikke lave nye tabeller, men lave et data join on-the-fly med SQL. Alternativt kunne vi oprette en ny tabel med de joinede tabeller.

1. Klik under "tables" og vælg skat_momsbetaling_2012
2. Paste denne SQL ind i SQL fanen:

```
SELECT kom.cartodb_id,kom.the_geom_webmercator,
kom.komnavn_moms,
skat.komnavn,
skat.omsaetning,
skat.branche
FROM kommune_union kom
LEFT JOIN skat_momsomsaetning_2012 skat ON (kom.komnavn_moms=skat.komnavn
  AND skat.branche='Murere')
  ORDER BY kom.komnavn_moms
```

1. Øvest i midten skiftes fra "DATA VIEW" til "MAP VIEW"
2. I højre side vælges fanen "wizards". Her vælges choropleth og bruges column=omsaetning

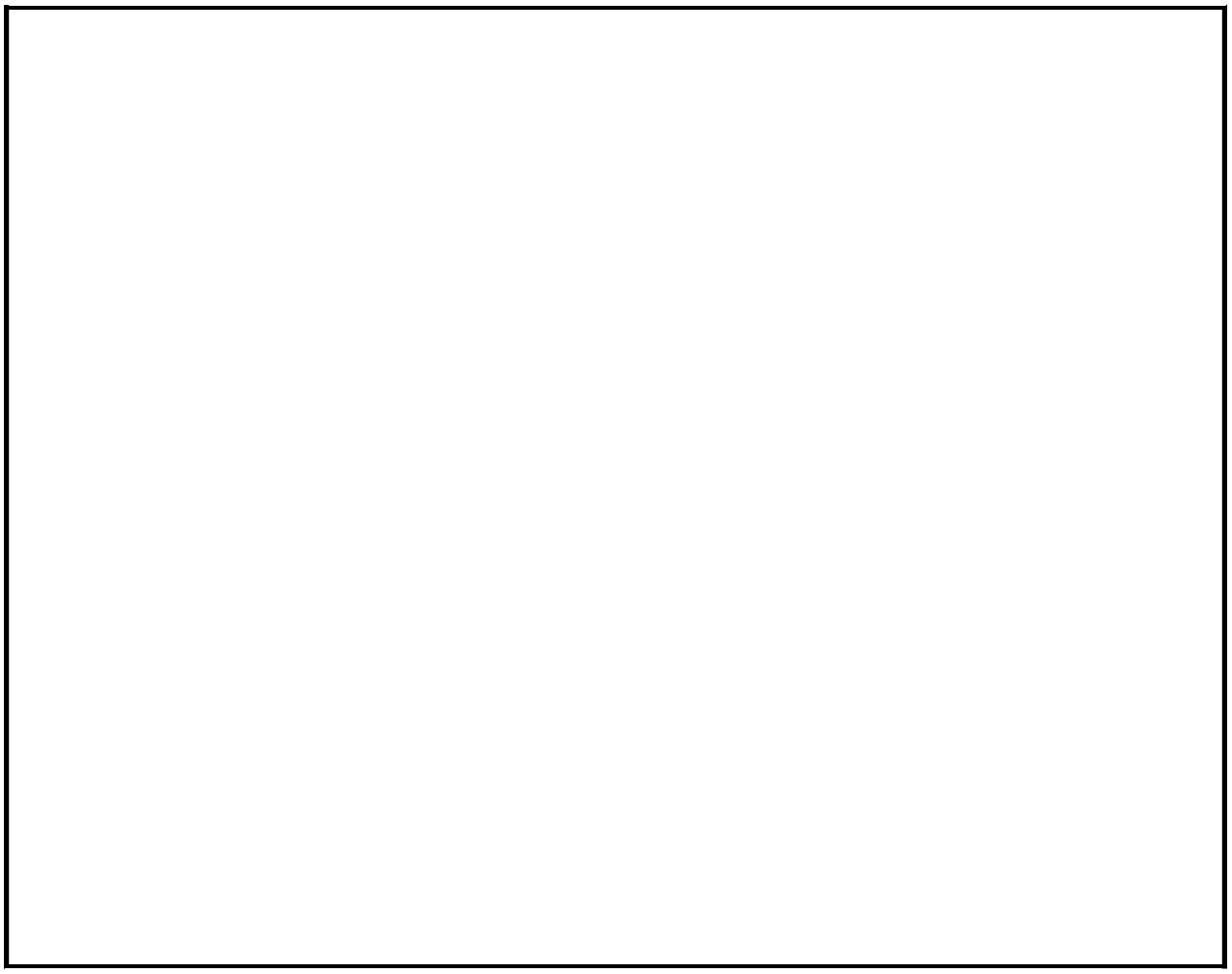
3. Vælg CSS fanen og tilføj en sort farve for null-værdierne (Læsø m.f) :

```
#skat_momsomsaetning_2012 [ omsaetning = null] {
  polygon-fill: #000000;
}
```

1. Vælg "legend" og tilføj titel
2. Vælg infovindue og konfigurér felter, der skal vises ved klik og hover (når man holder musen over et punkt/område)
3. Nederst i venstre hjørne vælges basemap og under "options"
4. Klik visualize og giv et navn
5. Klik "Share" og del kort som url, iframe eller javascript

> Bemærk, at vi henter kolonnerne cartodb_id og the_geom_webmercator. cartodb_id skal bruges til interaktioner i kortet som klik og hover the_geom_webmercator er en kolonne, der ikke vises, men bruges "under-the-hood" til rendering af kortet

Se hele gennemgangen her:



Arbejd selv videre

Prøv selv at lave kort med andre brancher, ved at ændre på branchen i SQL-fanen.

CVR Kortet

I denne CASE vil vi undersøge CVR datasættet og udarbejede et kort, som kan give os indsigt om hvilke brancher, der er fremherskende fordelt på kommuner og branchekoder.

--	--
Datasæt	CVR registret, kommunegrænser
Sværhedsgrad	Over middel
Forudsætninger	Installation af database på pc, Kendskab til SQL

CVR data

> For at gennemføre denne case kræves en solid erfaring med IT De beskrevne trin vises også som videoer

Derfor skal der installeres en PostgreSQL database samt PgAdmin (grafisk brugergrænseflade til PostgreSQL) før man kan følge de næste trin. Det letteste er at bruge en installationspakke som kan hentes her.

Når databasen og PgAdmin er installeret, kan de følgnede trin påbegyndes.

Indlæsning af CVR data i Postgres databasen

1. Hent CVR data
2. Opret en database med PgAdmin
3. Opret et skema med navnet cvr
4. Opret en tabel til indlæsning af CVR data

```
CREATE TABLE cvr.virksomheder
(
  modifikationsstatus character varying,
  cvrnr character varying,
  livsforloeb_startdato character varying,
  livsforloeb_ophoersdato character varying,
  ajourfoeringsdato character varying,
  reklamebeskyttelse character varying,
  navn_gyldigfra character varying,
  navn_tekst character varying,
  beliggenhedsadresse_gyldigfra character varying,
  beliggenhedsadresse_vejnavn character varying,
  beliggenhedsadresse_vejkode character varying,
  beliggenhedsadresse_husnummerfra character varying,
  beliggenhedsadresse_husnummertil character varying,
  beliggenhedsadresse_bogstavfra character varying,
  beliggenhedsadresse_bogstavtil character varying,
  beliggenhedsadresse_etage character varying,
  beliggenhedsadresse_sidedoer character varying,
  beliggenhedsadresse_postnr character varying,
  beliggenhedsadresse_postdistrikt character varying,
  beliggenhedsadresse_bynavn character varying,
  beliggenhedsadresse_kommune_kode character varying,
  beliggenhedsadresse_kommune_tekst character varying,
  beliggenhedsadresse_postboks character varying,
  beliggenhedsadresse_conavn character varying,
  beliggenhedsadresse_adressefritekst character varying,
  postadresse_gyldigfra character varying,
```

```
    postadresse_vejnavn character varying,
    postadresse_vejkode character varying,
    postadresse_husnummerfra character varying,
    postadresse_husnummertil character varying,
    postadresse_bogstavfra character varying,
    postadresse_bogstavtil character varying,
    postadresse_etage character varying,
    postadresse_sidedoer character varying,
    postadresse_postnr character varying,
    postadresse_postdistrikt character varying,
    postadresse_bynavn character varying,
    postadresse_kommune_kode character varying,
    postadresse_kommune_tekst character varying,
    postadresse_postboks character varying,
    postadresse_conavn character varying,
    postadresse_adressefritekst character varying,
    virksomhedsform_gyldigfra character varying,
    virksomhedsform_kode character varying,
    virksomhedsform_tekst character varying,
    virksomhedsform_ansvarligdataleverandoer character varying,
    hovedbranche_gyldigfra character varying,
    hovedbranche_kode character varying,
    hovedbranche_tekst character varying,
    bibranche1_gyldigfra character varying,
    bibranche1_kode character varying,
    bibranche1_tekst character varying,
    bibranche2_gyldigfra character varying,
    bibranche2_kode character varying,
    bibranche2_tekst character varying,
    bibranche3_gyldigfra character varying,
    bibranche3_kode character varying,
    bibranche3_tekst character varying,
    telefonnummer_gyldigfra character varying,
    telefonnummer_kontaktoplysning character varying,
    telefax_gyldigfra character varying,
    telefax_kontaktoplysning character varying,
    email_gyldigfra character varying,
    email_kontaktoplysning character varying,
    kreditoplysninger_gyldigfra character varying,
    kreditoplysninger_tekst character varying,
    aarsbeskaeftigelse_aar character varying,
    aarsbeskaeftigelse_antalansatte character varying,
    aarsbeskaeftigelse_antalansatteinterval character varying,
    aarsbeskaeftigelse_antalaarsvaerk character varying,
    aarsbeskaeftigelse_antalaarsvaerkinterval character varying,
    aarsbeskaeftigelse_antalinclejere character varying,
    aarsbeskaeftigelse_antalinclejereinterval character varying,
    kvartalsbeskaeftigelse_aar character varying,
    kvartalsbeskaeftigelse_kvartal character varying,
    kvartalsbeskaeftigelse_antalansatte character varying,
    kvartalsbeskaeftigelse_antalansatteinterval character varying,
    produktionsenheder character varying,
    deltagere character varying
```

```
  )
  WITH (
    OIDS=FALSE
    );
```

5.Indlæs data med COPY funktionen Udskift stien til filen, så den passer til placeringen af din egen fil.

```
COPY cvr.virksomheder
FROM '/Users/mbj/Downloads/39247470_42355_20150123134947/39247470_42355_20150123134947_VI
WITH DELIMITER ','
CSV HEADER;
```

6.Undersøg hvilken branche, der har flest virksomheder i hver kommune

```
SELECT t3.kom_kode,t3.beliggenhedsadresse_kommune_tekst,max(t3.hovedbranche_tekst),t3.ant

FROM
(select '0' || beliggenhedsadresse_kommune_kode as kom_kode, beliggenhedsadresse_kommune_
FROM cvr.virksomheder GROUP BY beliggenhedsadresse_kommune_tekst,beliggenhedsadresse_komm
) t3
JOIN
(
  select beliggenhedsadresse_kommune_tekst, max(antal) as antal
  from
  (
    select beliggenhedsadresse_kommune_tekst, hovedbranche_tekst, count(1) as antal
    FROM cvr.virksomheder GROUP BY beliggenhedsadresse_kommune_tekst , hovedbranche_tekst
    ) t
    GROUP BY beliggenhedsadresse_kommune_tekst
    ) t2
    ON (t2.antal = t3.antal AND t2.beliggenhedsadresse_kommune_tekst = t3.beliggenhedsadr
    GROUP BY t3.kom_kode,t3.beliggenhedsadresse_kommune_tekst,t3.antal
    ORDER BY beliggenhedsadresse_kommune_tekst
```

Når vi kigger på resultatet af denne forespørgsel, kan vi se at i de fleste kommuner er den største branche en af følgende:

Branchekode
Andre organisationer og foreninger i.a.n.
Uoplyst
Ikke-finansielle holdingselskaber
Udlejning af erhvervsejendomme

> De frivillige organisationer er den branchekode alle foreninger(ikke sportsklubber) skal lægge sig under, og derfor er der rigtig mange af dem. De ligger typisk der hvor deres advokat bor, altså den advokat som administrerer organisationen. De uoplyste er virksomheder, som endnu ikke betaler moms, det vil sige helt nystartede virksomheder, eller meget små(omsætning på under 50.000).

Det kan der være mange årsager til. Det er nogle meget brede kategorier og muligvis er virksomhederne placeret i disse i mangel af andre dækkende kategorier. Vi vælger at fjerne disse fra vores liste. Desuden får vi flere resultater pr. kommune, hvis der er brancher med samme antal virksomheder. Vi vil kun have nøjagtigt +en branche pr kommune. Derfor skal vi sortere en fra. Vi vælger - helt ukritisk - at beholde den branchekode, som alfabetisk har det største forbogstav med Max. Det kan gøres med følgende sql

```
SELECT t3.kom_kode,t3.beliggenhedsadresse_kommune_tekst,max(t3.hovedbranche_tekst),t3.ant

FROM
(select '0' || beliggenhedsadresse_kommune_kode as kom_kode, beliggenhedsadresse_kommune_
FROM cvr.virksomheder GROUP BY beliggenhedsadresse_kommune_tekst,beliggenhedsadresse_komm
) t3
JOIN
(
  select beliggenhedsadresse_kommune_tekst, max(antal) as antal
  from
  (
    select beliggenhedsadresse_kommune_tekst, hovedbranche_tekst, count(1) as antal
    FROM cvr.virksomheder WHERE hovedbranche_tekst NOT IN ('Andre organisationer og foren
    ) t
    GROUP BY beliggenhedsadresse_kommune_tekst
    ) t2
    ON (t2.antal = t3.antal AND t2.beliggenhedsadresse_kommune_tekst = t3.beliggenhedsadr
    GROUP BY t3.kom_kode,t3.beliggenhedsadresse_kommune_tekst,t3.antal
    ORDER BY beliggenhedsadresse_kommune_tekst
```

Nu ligner resultatet noget, der kan give et billede af de fremherskende brancher i kommunerne. Vi eksporterer nu resultatet til CartoDB.

7.Marker SQL i PgAdmin og vælg i meuen: Query-> Execute to file. Kald filen top_brancher.csv

8.Log ind på CartoDB og opret ny tabel som i CASE om momsdata og upload top_brancher.csv

9.Tilret kom_kode feltet:

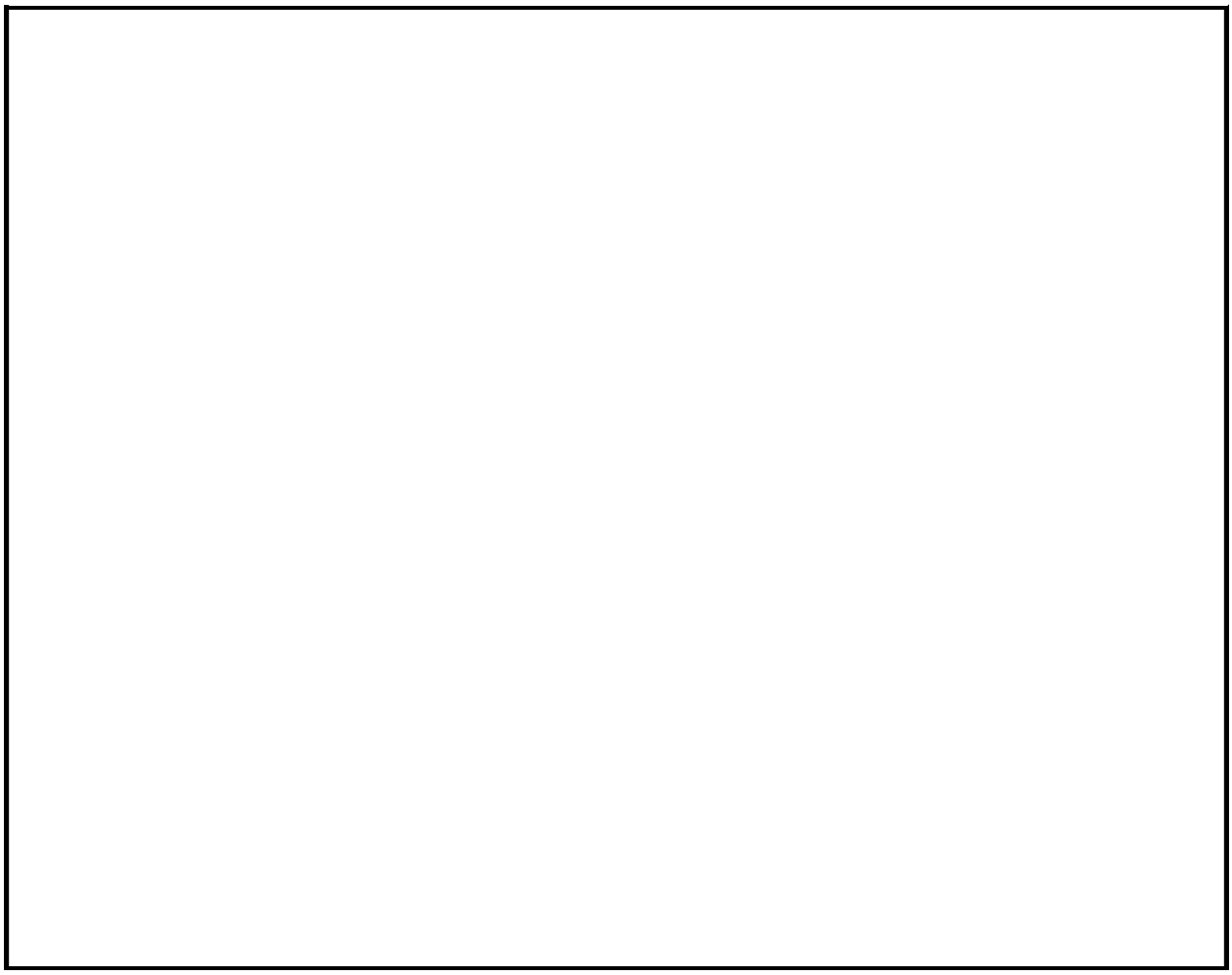
Vi vil nu joine vores analyseresultat med vores kommunegrænser. Kommunekoden i cvr er et heltal, mens kommunekoden i vores DAGI koommuner er et tekstfelt bestående af et foranstillet nul efterfulgt af kommunekoden. Derfor tilføjer vi en kolonne til vores upladede datasæt (top_brancher). Vi åbner vores tabel i CartoDB og vælger SQL fanen og indtaster følgende sql, som opretter en ny kolonne til vores kommunekoder:

```
ALTER TABLE top_brancher ADD column kom_kode_ny character varying;

UPDATE top_brancher SET kom_kode_ny = '0' || kom_kode
```

Vi er nu klar til at lave vores JOIN af vores cvr analyseresultat og vores DAGI kommunegrænser. Vi vil benytte en funktion (merge tables) i CartoDB til at JOINE to tabeller i næste kapitel.

Se video af workflowet i dette kapitel her:



Kort med CartoDB

For at visualisere resultatet i CartoDB skal vi lave et **merge** mellem to tabeller. Vi skal anvende kolonnerne med geometri fra kommunetabellen som skal **merges** med tabellen vi netop har uploaded. Det kan også udføres med et SQL JOIN, men denne gang prøver vi **merge** funktionen i CartoDB.

For at konstruere kortet skal følgende trin gennemføres.

> Alle trin gennemgås også i videoen nedenfor

1.Opret en kommunegrænse tabel med de nødvendige kolonner:

Vi har behov for en kolonne med kommunekoden fra kommunetabellen. I "Momskortet" JOINEDE vi geografi med momsdata på kommunenavnet. Denne gang har vi en kommunekode i top_branche tabellen og derfor er det hele lidt lettere.

Vi kan lave en ny kommunetabel med følgende sql (Se video nedenfor)

```
SELECT st_union(the_geom) as the_geom, komnavn,komkode  FROM kommune GROUP by komnavn,kom
```

Klik "Create table from query"

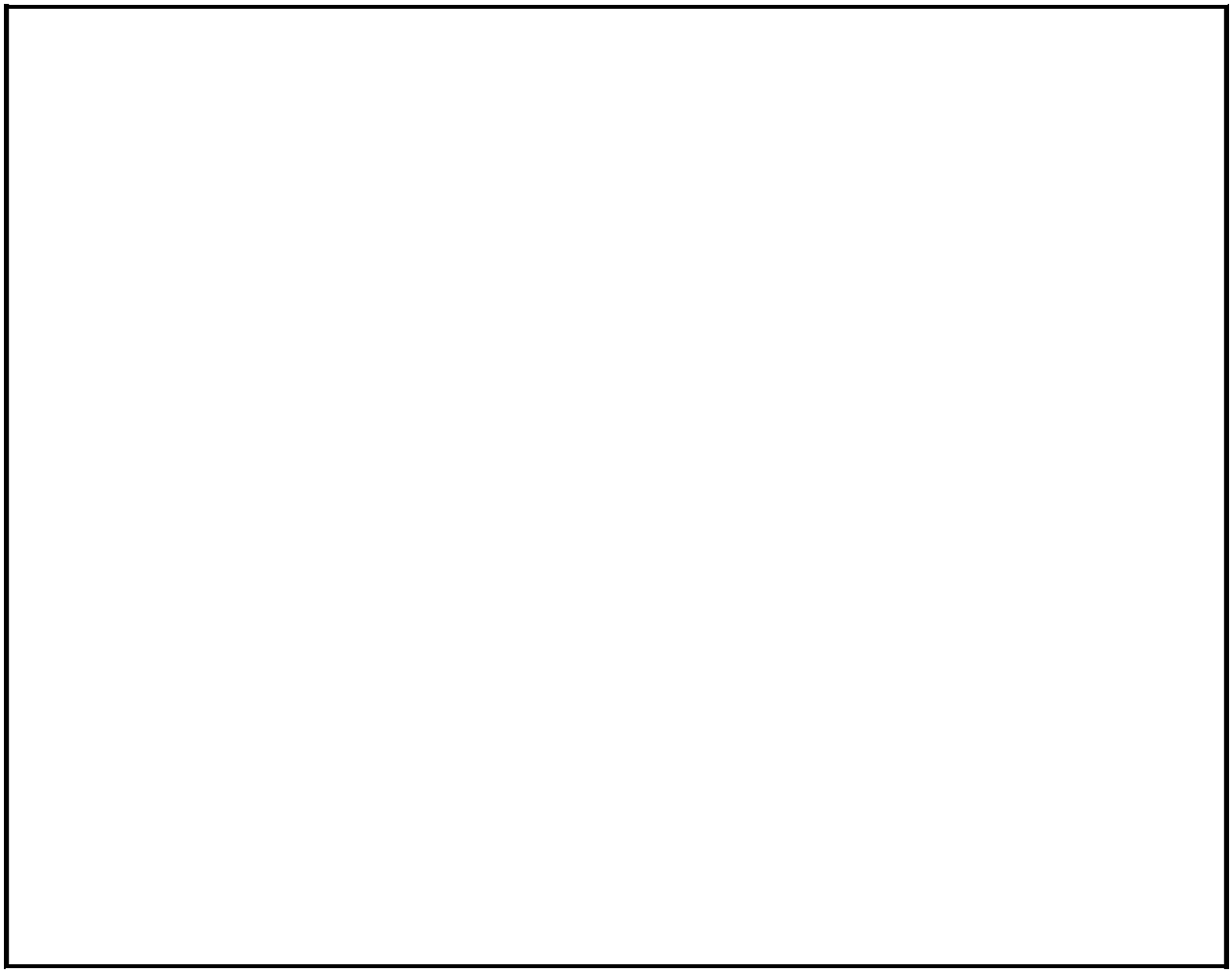
2.Merge tabellerne til ny tabel:

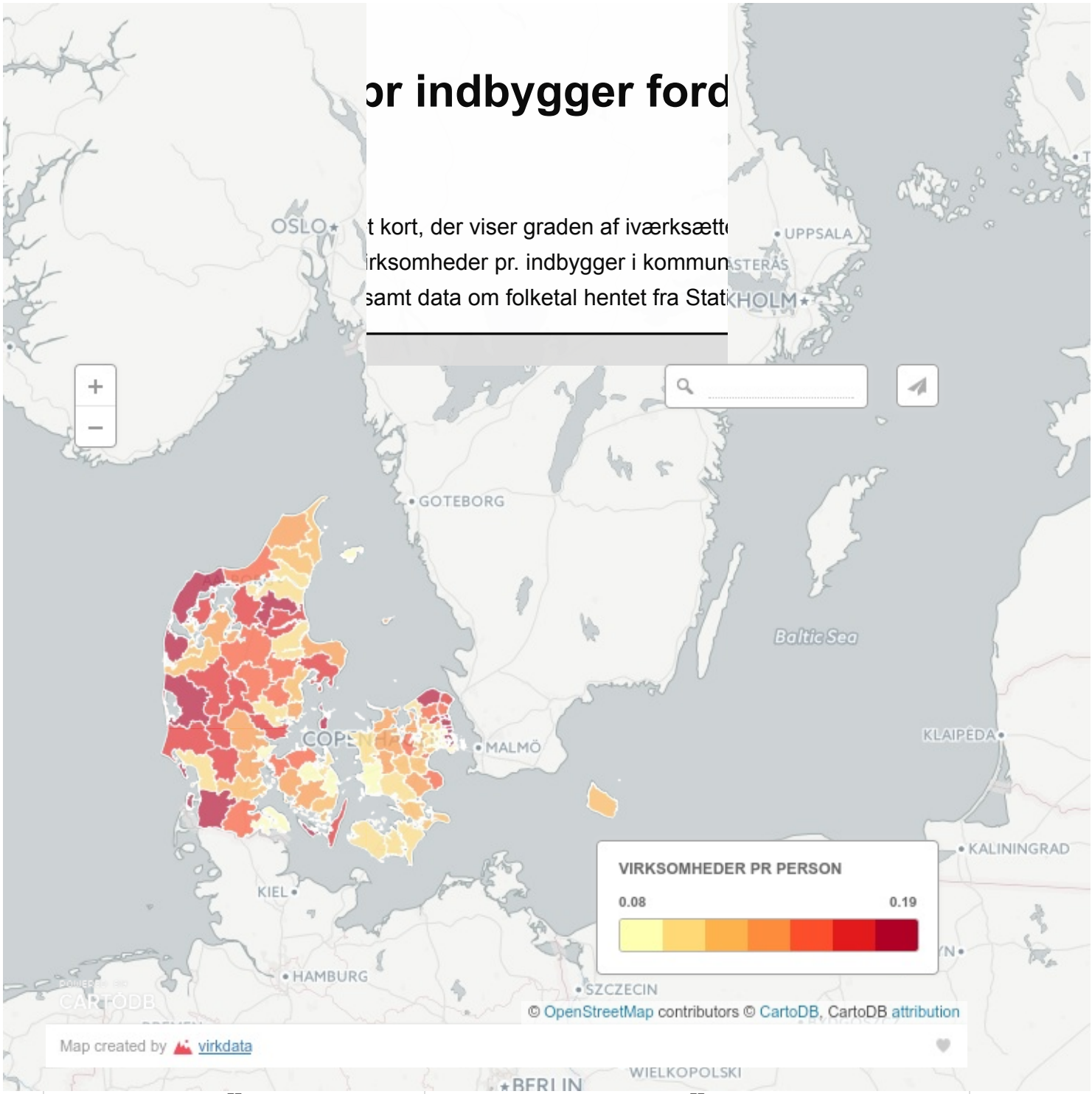
Vælg tabellen "top_brancher". Klik i menupunkt Edit->Merge with table Vælg den kolonne, der skal JOINES på (se video)

3.Visualisér tabellen:

Visalisér tabellen på samme måde som i Momskortet.

Se video af workflowet her:





--	--
Datasæt	CVR, kommunegrænser
Sværhedsgrad	Middel
Forudsætninger	Kendskab til SQL

Data til virksomheder pr. indbygger

For at svare på spørgsmålet om antallet af virksomheder pr. indbygger fordelt på kommune, har vi behov for at lave en sammentælling af virksomheder pr kommune.Da vi allerede har indlæst CVR data i vores lokale PostgreSQL database, kan vi let lave optællingen med følgende SQL:

> Alle trin gennemgås også i videoen nedenfor

Antal virksomheder fordelt på kommune

```
SELECT count(*), beliggenhedsadresse_kommune_kode
 FROM cvr.virksomheder
  GROUP BY beliggenhedsadresse_kommune_kode
   ORDER BY beliggenhedsadresse_kommune_kode
```

Vi får følgende (trunkerede resultat):

count	beliggenhedsadresse_kommune_kode
70947	101
11968	147
3594	151
2593	153
....	...

Dette resultat skal nu eksporteres til CartoDB som en ny tabel. Vi gør det på tilsvarende måde som i "Momskortet":

Marker SQL i PgAdmin og vælg i meuen: Query-> Execute to file. Kald filen cvr_count.csv.

Log ind på CartoDB og opret ny tabel som i CASE om momsdata og upload cvr_count.csv

> Hvis man har et unix-baseret (eller windows med WGET eller CURL) system kan man uploade i et skridt med SQL med dette hack:

```
COPY
(SELECT beliggenhedsadresse_kommune_kode, count(*) as antal
 FROM cvr.virksomheder
  WHERE beliggenhedsadresse_kommune_kode <> ''
   GROUP BY beliggenhedsadresse_kommune_kode
   ORDER BY antal DESC)
 TO PROGRAM 'cat <&0 > /tmp/data.csv; curl -F "file=@/tmp/data.csv;filename=cvr_coun
   DELIMITER ','
   CSV HEADER;
```

Folketal fra Danmarks Statistik

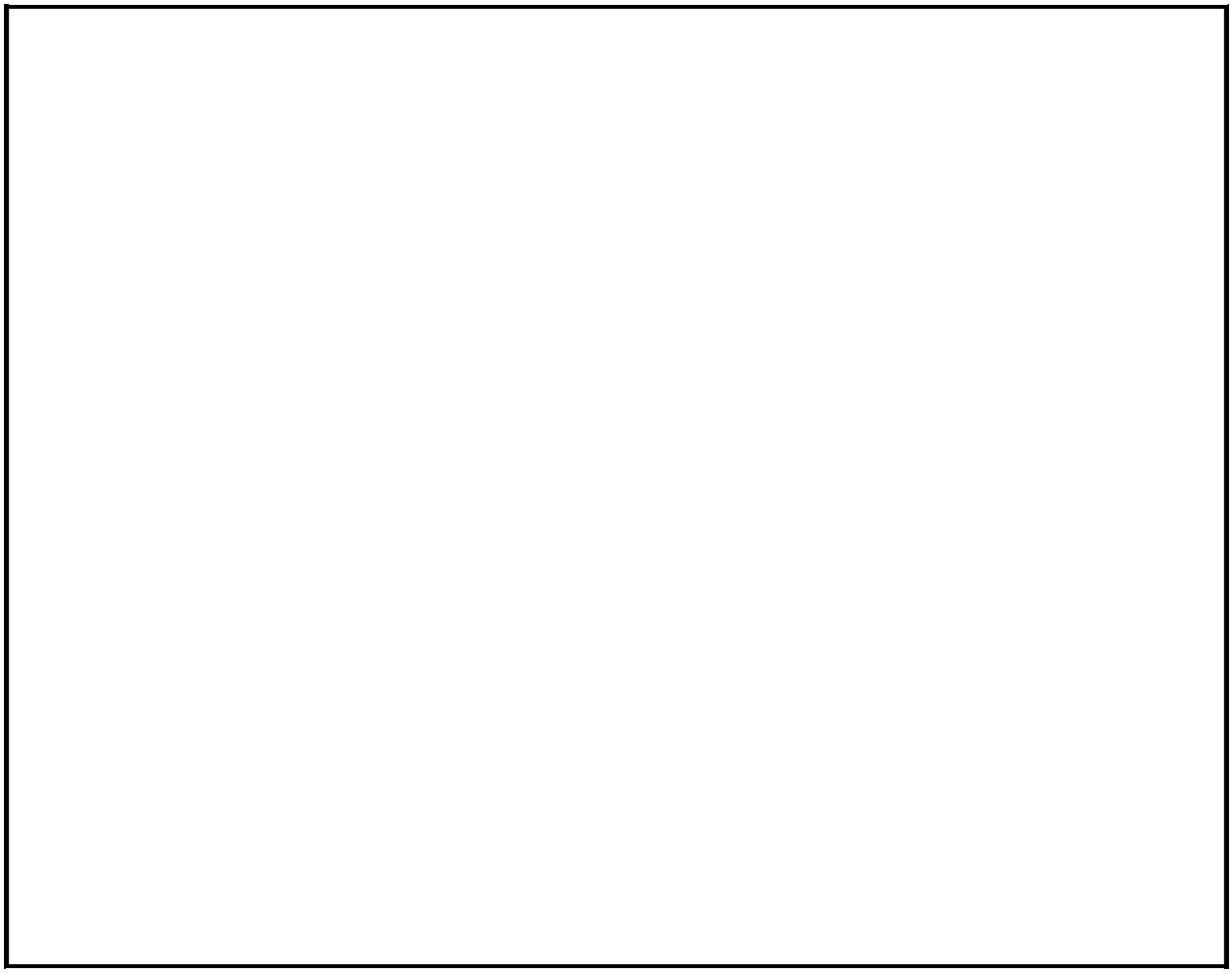
Danmarks Statistik har folketællinger fordelt på kommune. Man kan selv vælge sine kolonner og hvilket format man vil have data i. I denne øvelse anvender vi ikke denne løsning, men i stedet henter vi direkte ind i CartoDB med et API, hvor man kan konstruere sin egen forespørgsel mod Danmarks Statistik og dermed hente data direkte ind i CartoDB. Den forespørgsel vi skal bruge for at hente folketal med API'er ser således ud.

http://api.statbank.dk/v1/data/FOLK1/CSV?valuePresentation=Code&OMRÅDE=*&TID=2015K1

Log ind på CartoDB og opret en nye tabel og indsæt adressen ovenfor. Omdøb tabellen til **folketal** Dermed oprettes tabellen automatisk og det er ikke nødvendigt at uploade en fil.

> Du kan læse mere om Danmarks Statisks API her, samt lave andre forepørgsler.

Nu går vi videre til at danne selve kortet og resten af øvelsen foregår i CartoDB.



Kortet

Vi har nu to tabeller i CartoDB **cvr_count** og **folketal**

Vi skal nu joine de to tabeller på kolonnen med kommunekode. Det kan gøres med SQL, men vi vil gøre det med CartoDBs **merge** funktion.

Inden vi kan **merge** skal vi lige fjerne overflødige data fra tabellen **folketal**

> Alle trin gennemgås også i videoen nedenfor

I datasættet er der rækker med som ikke repræsenterer kommuner. Det er folketal fra regioner fra hele landet. Det er alle de rækker, hvor kolonnens område er mindre end 100.

Klik på tabellen **folketal** og åben SQL fanen i højre side.

```
SELECT * FROM folketal WHERE omrade < 100
```

Slet disse rækker med:

```
DELETE FROM folketal WHERE omrade < 100
```

Derudover skal vi foranstille et **0** så vi kan joine med tabellen **kommune_merge_kode** med kommunegrænser som vi oprettede i en tidligere øvelse.

Opret en nye kolonne:

```
ALTER TABLE folketal ADD column kom_kode_ny character varying;
```

Indsæt kommunekoden + et foranstillet "0":

```
UPDATE folketal SET kom_kode_ny = '0' || omrade
```

Klik på kolonnen **indhold** og omdøb den til **personer**.

Vælg tabellen **cvr_count** og omdøb på samme måde kolonnen **antal** til **virksomheder**.

Klik på menupunktet Edit>Merge with table.

Vælg **column join**.

I venstre panel vælges kolonnen **beliggenhedsadresse_kommune_kode**. Det er den kolonne vi vil joine med fra **cvr_count** tabellen.

I højre side vælges tabellen **folketal** og vælg kolonnen **omrade**. Dermed joiner vi de to tabeller på kommunekoden.

Navngiv tabellen **cvr_count_folketal_merge**.

Åben nu tabellen **cvr_count_folketal_merge** og vælg igen Edit>Merge with table.

Vælg at joine med tabellen **kommune_merge_kode** på kolonnerne **kom_kode_ny** og **komkode**.

Navngiv den nye tabel eksempelvis **virk_pr_person**.

Nu opretter vi en ny kolonne, der viser, hvor mange virksomheder, der findes pr. person fordelt på kommune.

Opret kolonnen:

```
ALTER TABLE virk_pr_person ADD column virk_person double precision;
```

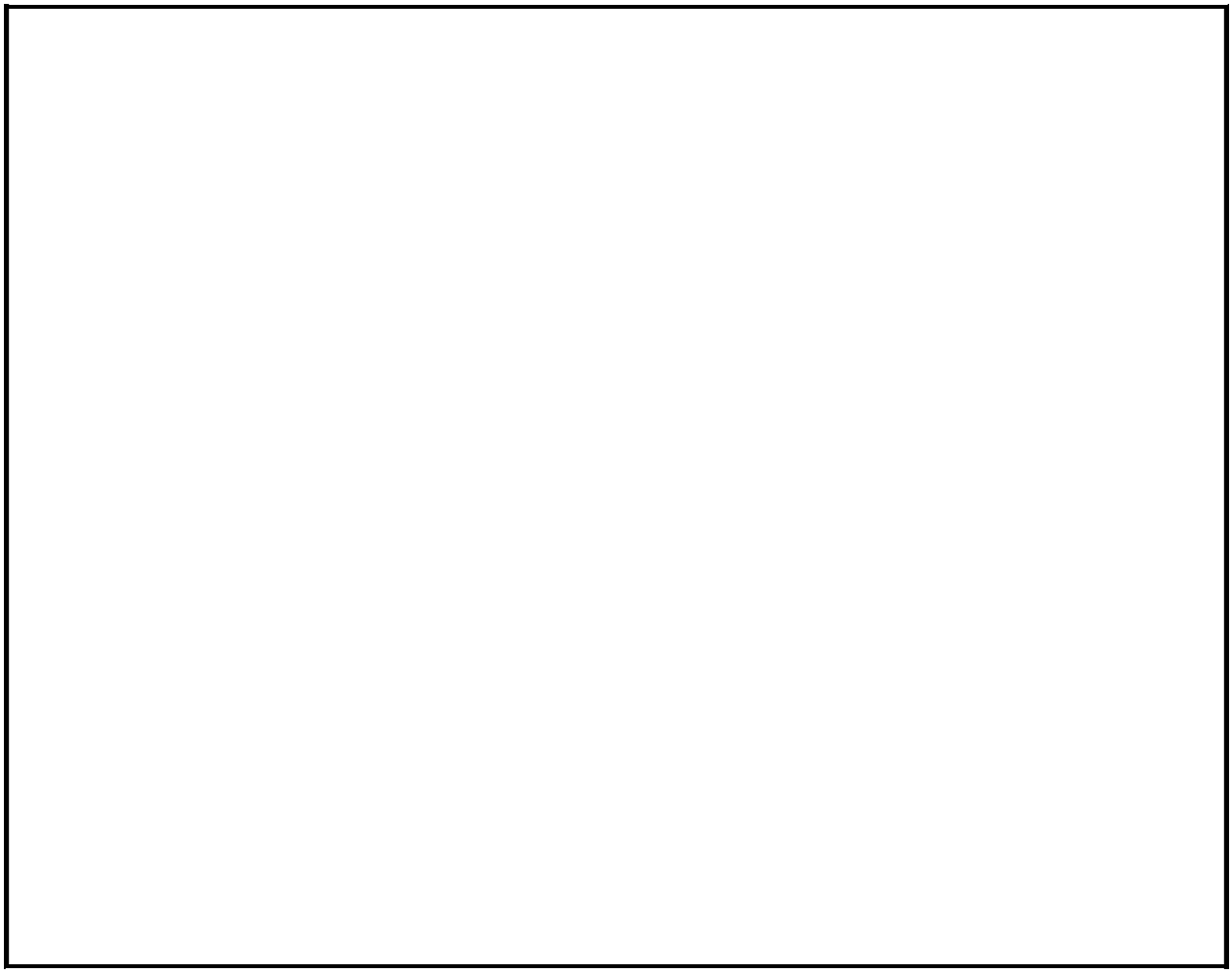
Opret beregnet kolonne med ny værdi:

```
UPDATE virk_pr_person SET virk_person = virksomheder/personer::double precision
```

Nu har vi alle data så vi kan lave kortet i CartoDB, tryk på fanen **map View** i toppen.

Klik på fanen **Wizards** og vælg **choropleth**. Vælg kolonnen **virk_person** vi netop har oprettet.

Vælg også **Legends** og vælg en titel f.eks. **Virksomheder pr. person**



Kort med d3

D3 er et javascript bibliotek, der primært anvendes til at lave visualiseringer til webbrowsere. D3 har god understøttelse for at producere kort. Nedenstående er et eksempel på det foregående kort, men udviklet med D3 og SVG elementer. Datasættet er det samme og hentes direkte fra CartoDBs SQL API.

Forsøg selv at lave en html-fil fra følgende kodeeksempel. Og indsæt et link til din egen cartodb-bruger.

Se Eksempel her Prøv også at zoome med scroll hjulet.

```
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html;charset=utf-8">
    <title>Kort med d3</title>
    <script src="http://d3js.org/d3.v3.min.js"></script>
    <script src="http://d3js.org/d3.geo.projection.v0.min.js"></script>
    <script src="http://d3js.org/topojson.v0.min.js"></script>

    <style type="text/css">

    body{
        background:white;
    }
    svg {
      width: 960px;
      height: 500px;
      background: none;
    }
    svg:active {
      cursor: move;
      cursor: -moz-grabbing;
      cursor: -webkit-grabbing;
    }
    .globe {
      fill: black;
      fill-opacity: 1;
      stroke: #111;
      stroke-width:1px;
    }
    #denmark path {
      stroke: #333;
      stroke-linecap: round;
      stroke-linejoin: round;
    }
```

```
    </style>
  </head>
  <body>
    <h1>Virksomheder pr. person</h1>
    <script type="text/javascript">

    var denmark = 'denmark';


    var svg = d3.select("body")
            .append("svg")
            .call(d3.behavior.zoom()
                .on("zoom", redraw))
            .append("g");

            // Define the color scale for our choropleth
    var fill = d3.scale.linear()
            .domain([0.0841347229055777,0.0915873241438113, 0.0995412160259702, 0.1050053
            .range(["#FFFFB2","#FED976","#FEB24C", "#FD8D3C", "#FC4E2A", "#E31A1C", "#B10


    // Our projection.
    var xy = d3.geo.mercator()
              .scale(4500)
              .center([11,56.3]);

    var path = d3.geo.path()
                .projection(xy)

    svg.append("g").attr("id", "denmark");

    //sql.execute("SELECT ST_Simplify(the_geom,0.01) as the_geom, pop2005 as population F

    d3.json("http://virkdata.cartodb.com/api/v2/sql?q=SELECT the_geom, virk_person FROM v
      //var subunits = topojson.object(collection);
      var tj = {kommuner: {type: "GeometryCollection", geometries: []}};
      for (i in collection.objects){
        tj.kommuner.geometries.push(collection.objects[i])
      }
      collection.objects = tj

      svg.select("#denmark")
        .selectAll("path")
        .data(topojson.object(collection, collection.objects.kommuner).geometries)
        .enter().append("path")
        .attr("fill", function(d) { return fill(d.properties.virk_person); })
        .attr("stroke-width", "0.1px")
        .attr("fill-opacity", "1")
        .attr("d", path.projection(xy));
            });

    function redraw() {
      svg.attr("transform", "translate(" + d3.event.translate + ")scale(" + d3.event.scal
```

```
    }

    </script>
  </body>
</html>
```

Læs mere om d3

Bidrag selv til bogen

Fork selv bogens repository på Github og lav Pull requests. Virkdata vil merge relevante pull requests som automatisk vil indgå i denne bog.

Bogen er skrevet med gitbook og kræver NodeJS er installeret.

Når NodeJS er installeret kan gitbook og afhængigheder installeres på følgende måde:

Installér gitbook

```
npm install gitbook -g
```

Installér andre krævede nodeJS moduler

```
npm install
```

Man kan rette i bogen lokalt på eget EDB anlæg med en live reload server, der automatisk opdaterer browseren, når filerne ændres:

```
gitbook serve
```

Derefter kan man se bogen på: http://localhost:4000/

Når de ønskede ændringer er udarbejdet sendes et Pull request til

https://github.com/virkdata/opendataschool