# Indholdsfortegnelse

[Introduktion 1](#_Toc195261814)

[Første del: Extract 1](#_Toc195261815)

[Hent data fra CSV-filer 1](#_Toc195261816)

[Hent data fra SQL-serveren 2](#_Toc195261817)

[Hent data fra API 2](#_Toc195261818)

[Anden del: Transform 3](#_Toc195261819)

[CSV-filer 3](#_Toc195261820)

[SQL-serveren 4](#_Toc195261821)

[Modelvisning 4](#_Toc195261822)

[Data i Power BI 4](#_Toc195261823)

[Generelle bemærkninger 5](#_Toc195261824)

# Introduktion

Da jeg ikke har tidligere erfaring med programmering, har mit primære fokus i denne opgave været at tage tingene skridt for skridt, og forstå hvordan de enkelte elementer fungerer, frem for at lave et færdigt produkt.

På grund af min manglende erfaring med programmering, har jeg fået Chat GPT til at hjælpe mig med en trin-for-trin guide, hvor den fortæller mig hvad jeg skal gøre, men jeg så vidt muligt selv skriver koden.

# Første del: Extract

I denne fase har jeg kigget på, samt ladet mig inspirere af hvad jeg har lavet i .csv opgaven i undervisningsuge 2 (UU2), samt i SQL-opgaven i undervisningsuge 4 (UU4).

## Hent data fra CSV-filer

Jeg har startet med at tage koden fra Kickstart kode dokumentet, og taget udgangspunkt i dette. Jeg har dog valgt at importere/bruge **pandas** i stedet for **polars**, da jeg dels har arbejdet med pandas i UU2, dels kan forstå at pandas er mere begyndervenligt.

Fredag d. 4/4 havde jeg et møde med Niels og Christoffer fra Århus, der hjalp mig videre i CSV-delen af opgaven. Jeg har ydermere brugt GitHub Copilot inde i Visual Studio Code til at hjælpe mig med at fikse fejl og mangler i koden.

Derudover har jeg skrevet en del kommentarer i selve koden.

## Hent data fra SQL-serveren

Jeg har taget en del af den kode, jeg lavede i forbindelse med UU4 opgaven, og genbrugt denne.

Jeg har været i tvivl om hvorvidt vi skulle bruge den SQL-datafil, der ligger i *Data DB* mappen, eller på den eksterne server. Efter en korrespondance med Christoffer, fandt jeg ud af at vi skal bruge den SQL database, der ligger på den eksterne server.

For at finde ud af hvilken database og hvilke tabeller, jeg skal bruge, har jeg gjort følgende:  
1) Genbrugt den kode, jeg lavede i UU4 (jeg har brugt Chat GPT til at guide mig til hvordan jeg logger ind på den eksterne server), til at logge ind på SQL serveren, og hente en oversigt over databaser, der giver følgende resultat:

- default

- information\_schema

- mysql

- performance\_schema

**- productdb**

- sys

Hvor jeg skal bruge **productdb**.

2) Dernæst har jeg ligeledes genbrugt noget kode fra UU4, der kigger efter hvilke tabeller der er i denne database. Der giver følgende resultat:

- brands

- categories

- products

- stocks

Jeg har nu en kode, der kan hente data fra disse 4 tabeller.

## Hent data fra API

Jf. den trin for trin guide, jeg har fået fra ChatGPT, er jeg startet med at gå ind i Swagger UI ved at skrive <http://192.168.20.171:8000/docs> i min browser.

Dernæst har jeg installeret filerne fra *requirements.txt* filen, ved at skrive *pip install -r requirements.txt* i terminalen.

Jeg har fået meget hjælp fra ChatGPT til denne del af opgaven, og jeg er løbet ind i en del fejl, som jeg har brugt ChatGPT til at hjælpe mig med at løse. Der var en del udfordringer med formateringen, og der skulle en del kodeændringer til, før at resultatet kom overskueligt ud som ønsket, og ikke bare som én lang tekststreng.

# Anden del: Transform

Jeg har oprettet en ny fil til Transform-koden.

## CSV-filer

Jeg har åbnet selve CSV-filerne, og transformeret data’et i Excel, for at danne mig et overblik over datarenheden i disse. Da begge filer er små og overskuelige, har jeg altså kigget på hele datasettet. Havde de været større, f.eks. hundredevis af rækker eller derover, havde jeg blot lavet nogle stikprøver, samt sorteret nogle af kolonnerne for at finde uregelmæssigheder.

Ud fra disse tjek, har jeg gjort mig følgende observationer og overvejelser:

**Staffs:**

* Data er som udgangspunkt meget rent og konsistent, så dokumentet behøver ikke den store transformering
* Dog har jeg bemærket, at der mangler et *manager ID* på én af medarbejderne. Måske har denne medarbejder ikke en leder, eller måske *er* vedkommende lederen? Dette løser jeg ved at ændre *null*-værdier til 0, således at chefer/medarbejdere uden en leder ”skiller sig ud” og kan sorteres/fremsøges separat.
* Derudover er *active* skrevet som et tal. Dette kan med fordel ændres til en true/false funktion (boolean), hvilket jeg vil gøre
* Desuden har medarbejderne ikke en unik nøgle-id – og selvom alle navne i dette tilfælde *er* unikke, vil det stadig være en god idé at tildele hver medarbejder et unikt ID, i stedet for at bruge navnet, f.eks. hvis der kommer en ny medarbejder, der har samme for- og efternavn som en eksisterende medarbejder.
* Sluttelig har jeg besluttet mig for at slå for- og efternavne sammen i én kolonne
* Og til sidst vil jeg køre et data-rens på mails, for at tjekke for, samt slette evt. mellemrum

**Stores:**

* Dette ark er ligeledes meget rent, og har i øvrigt kun 3 rækker.
* Man kunne godt overveje at give de tre butikker hver deres unikke nøgle-id. Men for at gøre *staffs* arket mere læsevenlig, vil jeg beholde butikkernes egne navne som nøgle-id, og ligeledes bruge butikkerne i *staffs* som sekundær nøgle.
* Derudover vil jeg ligeledes køre et data-rens på mails, for at tjekke for, samt slette evt. mellemrum
* Efterrationalisering: Da der er en del andre dataark, der har en kolonne der hedder *store\_name* (f.eks. staffs og stocks), mens den i stores bare hedder *name*, kunne jeg med fordel ændre denne kolonneoverskrift til *store\_name*, for at holde navngivningen konsistent, og for nemmere ”sammenkobling” i modelvisningen i f.eks. Power BI.

Jeg har brugt Chat GPT til at hjælpe mig med at skrive koden.

## SQL-serveren

Også her virker data’en meget ren, og jeg vil overveje at holde mig til ”best practices” rens, som at sikre at tal er formateret som tal. Derudover vil jeg også overveje at:

* **Products:**
  + Formatere *model\_year* som date (år)
  + Formatere *list\_price* som beløb (USD)

**Jeg nåede kun hertil i programmeringen**

# Modelvisning

I Power BI’s Modelvisning har jeg ”tegnet” en sammelkobling imellem de forskellige databaser, med hvordan de kan interagere med hinanden.

Et eksempel på et flow kunne være:

En kunde i *customer* lægger en ordre i *orders*. *Orders* er koblet op på *order\_items*, hvor man kan se hvilke varer, kunden har købt (*product\_id*). Dette *product\_id* er koblet op på *products*, hvor man så kan se det elle de konkrete produkter, kunden har købt (jeg bemærker dog, at databasen vist er bygget op sådan, at hver kunde kun kan købe/har købt én ting. På den ene side giver det mening, da man nok kun vil købe én cykel. På den anden side misser vi køb af ekstraudstyr (f.eks. cykelhjelm, lygter, opgradering af dæk m.m.) på denne måde). I *products* kan man så gå videre til *stocks*, og se, om den enkelte butik faktisk har det produkt, kunden har bestilt.

# Data i Power BI

Jeg har (torsdag d. 10/4) brugt dagen på at hoppe direkte til databearbejdning i Power BI, hvor jeg har importeret CSV-versionerne af alt data’en, og udvalgt følgende ting at præsentere:

* Et overblik over samlet antal medarbejdere samt samlet lagerbeholdning
* Detaljeret lagerbeholdning på baggrund af butik
* Månedlig omsætning fordelt på de tre butikker

Mit primære fokus har været ETL-programmeringsdelen, hvorfor jeg ikke har gået i dybten med Power BI diagrammet denne gang. Men her er nogle overvejelser jeg kunne gå videre med, såfremt jeg skulle gå i dybten med selve dataen:

* Salg over tid på baggrund af kategorier: Er der markante udsving i salget over tid inden for de forskellige kategorier? I så fald, kan dette bruges til at tilpasse sig markedet, ved f.eks. at op- eller nedskalere på forskellige cykeltyper? Her kan man også gå ned og analysere på de enkelte butikker
* Kundeforhold: Er der kunder, der har handlet flere gange? Køber flere og/eller dyrere cykler? Dette kan bruges til at målrette salgskommunikation, f.eks. i de nyhedsbreve, der sendes ud. Når der er opsat regler for kategorier (f.eks.: køber dyre cykler, køber også børnecykler osv.), og hvis virksomheden bruger MailChimp, kan MailChimp kobles op på virksomhedens netbutik og automatisk tildele kategorier til nye kunder, der så får målrettede nyhedsbreve.
* Flow i lager: Da alle 3 butikker har enormt mangle cykler på lager – over 4000 hver – kan vi vejlede virksomheden i lagerstyring, ved at kigge på hvor mange af hver cykel de sælger, sammenholdt med aktuel lagerbeholdning. Er der f.eks. cykler der har stået lang tid på lager, uden at blive solgt, kan man overveje at lade være med at indkøbe flere af disse. Er der omvendt en cykel, man sælger mange af, *og* samtidig har mange af pågældende cykel på lager, kunne man enten overveje at bibeholde nuværende indkøbsflow fra producenten, eller måske endda opskalere indkøb af disse modeller, for at opnå større rabatter hos producenten, nu når man allerede har et højt flow på disse. Omvendt, så kunne man overveje at købe færre cykler af de modeller, man sælger få af (har mindre flow af), så de ikke hober sig op. Jeg kunne godt forestille mig, at butikkerne kunne halvere deres lagerbeholdning, og dermed spare lagerplads samt gøre det nemmere for medarbejderne af finde de cykler, der skal sælges.

# Generelle bemærkninger

* Jeg har brugt Chat GPT til at lave en trin for trin guide til mig. Jeg har startet med at få den til at lave opgaven helt, for at få en fornemmelse af slutresultatet, og har derefter bedt den om at guide mig igennem trin for trin.
* IP-adressen til API/SQL-serveren ændres jævnligt. Hvis denne kode skal i ”prod” med dette forhold, skal der laves et modul der henter den aktuelle IP-adresse, hvis man vil undgå at en person skal skaffe IP-adressen manuelt hver gang den ændres.