Sebastian Sigurd Sandby Anthonsen,
Stephanie Skjøtt Rone &
Vilhelm Yunus Altintas

**Brief introduction & description of ONE of the dataset - Spotify**

- **How is the data structured? (E.g. Highligts of: Examples of data scales, dataset description, examples of data types)**

**Spotify**

Dataen er struktureret i en tabel med 18 kolonner, som viser information om populære sange på Spotify fra 2000-2019. I datasættet er der forskellige datatyper og skalaer. Den mest forekommende datatype er tal, men der findes også enkelte tilfælde af strings/ord samt boolean. Der findes nominal, interval og ratio i datasættet.

Datatyper:
**Strings/ord**: Sangtitler, kunstner og genre - hvor værdien består af ord og navne
**Number/tal:** fx. popularitet, udgivelsesår eller tempo
**Boolean**: Explicit og Mode, fordi de begge handler om to mulige tilstande (Ja/Nej, Sandt/Falsk) - som hyppigt bruges som 1 og 0.
**NULL**: Datasættet indeholder ingen NULL-værdier, da der ikke er manglede data i tabellen.

Data skalaer:
**Nominal** - ukategoriserede data; fx. *songs* eller *artist*, da de repræsenterer kategorier uden nogen naturlig rækkefølge.
**Ordinal** -Der er ingen ordinale variable i datasættet, fordi variablerne enten mangler naturlig rækkefølge (nominale) eller har både rækkefølge og meningsfulde afstande (interval/ratio), i modsætning til ordinalskalaen, hvor kun rækkefølgen giver mening, men ikke afstanden mellem værdierne.
**Interval**: fx *year* - der er en rækkefølge, og intervallerne er konstante, men der ikke nødvendigvis er et absolut nulpunkt (f.eks. år 2000 - 2001 er præcist samme afstand som 2019 - 2020).
**Ratio**: fx *danceability* (fra 0.0 til 1.0), hvor værdien har et naturligt nulpunkt (0.0 betyder slet ikke dansbar), og forskellene er meningsfulde.

**For each question below of BOTH of the datasets:**
a) Write a query solving the questions
b) Describe - how does the query work?

## Spotify

- How many songs have explicit content?

Sebastian Sigurd Sandby Anthonsen,
Stephanie Skjøtt Rone &
Vilhelm Yunus Altintas

```
SELECT song
FROM spotify
WHERE explicit > 0;
```

**SELECT:** Denne del siger, at vi vil hente kolonnen *song*
**FROM:** Her fortæller vi, at vi henter data fra tabellen, der hedder spotify.
**WHERE:** Denne betingelse filtrerer resultaterne, så vi kun får sange, hvor kolonnen *explicit* er større end 0.

**Resultat: 551**

- Display all songs made between 2000-2010

```
SELECT song
FROM spotify
WHERE year between 2000 and 2010
```

**SELECT:** Henter kolonnen *song*
**FROM:** Henter data fra tabellen *spotify*.
**WHERE:** Filtrerer resultaterne, så kun sangens udgivelsesår med et *year* mellem 2000 og 2010 (inklusive begge år) vises.

**Resultat: 1000 rækker**

- What is the name of the most danceable trac

| song | year | danceability |
|---|---|---|
| 4 My People (feat. Eve) | 2001 | 0.969 |

```
SELECT song, year, danceability
FROM spotify
WHERE year = "2001"
ORDER BY danceability desc
LIMIT 1;
```

**SELECT:** Henter kolonnerne *song, year* og *danceability*
**FROM:** Henter data fra tabellen *spotify*.
**WHERE:** Filtrerer resultaterne, så kun sange fra året *2001* medtages.
**ORDER BY:** Sorterer resultaterne efter kolonnen *danceability* i faldende rækkefølge (desc) - altså fra mest *danceable* til mindst *danceable*.
**LIMIT:** Begrænser resultatet til kun at vise den allerøverste række (den mest *danceable* sang).

- Display only the artist name and song title of tracks with a key between 1-5

```
SELECT `key`, song, artist
FROM spotify
WHERE `key` between 1 and 5
```

**SELECT:** Henter kolonnerne *key, song, artist*
**FROM spotify:** Henter data fra tabellen *spotify*.
**WHERE:** Filtrerer resultaterne, så kun sange med *key between* 1 og 5 (inklusive både 1 og 5) vises.
- *Key* skal være med backticks, da det er et reserveret ord i SQL

**Result: 793**

- Display all data of songs starting with 'B' that has a key between 1-5

```
SELECT `key`, song
FROM spotify
WHERE `key` BETWEEN 1 AND 5
AND song LIKE 'B%'
```

**SELECT:** henter to kolonner: *key* og *song*
**FROM:** henter data fra tabellen *spotify*.
**WHERE** filtrerer**,** så kun sange med toneart (key) mellem 1 og 5 inkluderes.
**AND:** Yderligere filter, så kun sange med titler, der starter med bogstavet *B*, medtages. Her bruges
**LIKE:** bruges til mønstergenkendelse.
**'B%':** betyder**,** at sangtitlen skal starte med *B* og kan have hvad som helst efter (procenttegnet % er wildcard).

**Resultat: 62 rækker**

## Coffee

- Display all data from customers who do not have a phone number entered into the customer table

```
SELECT*
FROM customer
WHERE phone_number IS NULL
```

| customer_id | firstname | lastname | gender | phone_number |
|---|---|---|---|---|
| 12 | Harry | Johnson | M | NULL |
| 14 | John | Taylor | M | NULL |
| 22 | Jennifer | NULL | F | NULL |
| NULL | NULL | NULL | NULL | NULL |

**SELECT**: Henter data fra alle kolonner fra tabellen.

**FROM**: Henter data fra tabellen *customer*
**WHERE:** Filtrerer de rækker, hvor *phone_number IS NULL* - altså hvor kunder ikke er registreret noget telefonnummer.

- Display all products and their country of origin in a single table

```
SELECT name, country
FROM
product
INNER JOIN country
USING
(country_id)
```

| name | country |
|---|---|
| Espresso | Columbia |
| Latte | Columbia |
| Cappuchino | Columbia |
| Black | Sri Lanka |
| Te | Sri Lanka |
| Black Exotic | Sri Lanka |
| Saft | Sverige |

**SELECT:** henter vi to kolonner; *name* (produktets navn fra *product*), og *country* (landets navn fra *country*)
**FROM:** henter data fra *product*, som er vores udgangspunkt.
**INNER JOIN:** kombinere med tabellen *country*, som har informationerne om landets navn.
**USING:** Her kan vi bruge *USING*, da begge tabeller har en kolonne med samme navn - i dette tilfælde *country_id*.

- Display all orders, customer firstname & lastname, and the country of origin of the product they ordered

```
SELECT o.order_table_id, c.firstname,
c.lastname,
p.name AS product_name,
co.country AS product_origin
FROM order_table o
INNER JOIN customer c
ON o.customer_id = c.customer_id
INNER JOIN product p
ON o.product_id = p.product_id
INNER JOIN country co
ON p.country_id = co.country_id;
```

| order_table_id | firstname | lastname | product_name | product_ori... |
|---|---|---|---|---|
| 1 | Chris | Martin | Espresso | Columbia |
| 2 | Chris | Martin | Latte | Columbia |
| 3 | Chris | Martin | Cappuchino | Columbia |
| 4 | Emma | Law | Espresso | Columbia |
| 5 | Chris | Martin | Espresso | Columbia |
| 6 | Mark | Watkins | Cappuchino | Columbia |
| 7 | Mark | Watkins | Black | Sri Lanka |
| 8 | Daniel | Williams | Te | Sri Lanka |
| 9 | Emma | Law | Te | Sri Lanka |
| 10 | Emma | Law | Te | Sri Lanka |
| 11 | Mark | Watkins | Te | Sri Lanka |
| 12 | Daniel | Williams | Cappuchino | Columbia |
| 13 | Emma | Law | Cappuchino | Columbia |

**SELECT:**
*o.order_table_id viser* ordrenummeret. *o.order_table_id* henter *ordrenummeret. c.firstname og c.lastname* henter kundens for- og efternavn. *p.name AS product_name* henter navnet på produktet. *co.country AS product_origin* viser landets navn, hvor produktet stammer fra. Vi giver begge  alias, så det bliver lettere at referere til i resultatet.

**FROM:** Her starter vi med at hente data fra tabellen *order_table. o* er et alias for *order_table,* som gør det lettere at referere til tabellen i resten af forespørgslen.

**INNER JOIN (custumer):** INNER JOIN customer c kobler hver ordre til en kunde via customer_id.
- Vi bruger *customer_id* til at forbinde de to tabeller. Det betyder, at vi henter informationer om kunden (fornavn og efternavn) for den specifikke ordre.
- *c* er et alias for tabellen *customer*.Så for hver ordre vil SQL'en finde den tilknyttede kunde (baseret på *customer_id*) og hente kundens fornavn og efternavn.

**INNER JOIN (product):** Vi gør endnu en *INNER JOIN* mellem *order_table* og *product:*
- Vi bruger *product_id* til at forbinde de to tabeller. Dette betyder, at vi henter oplysninger om produktet, som kunden har bestilt (produktnavn).
- *p* er et alias for tabellen *product*. Så for hver ordre vil SQL'en finde det specifikke produkt, som er blevet bestilt, ved at matche *product_id*.

**INNER JOIN (country)**: Vi laver en sidste ***INNER JOIN*** mellem *product* og *country*:
- Vi bruger *country_id* til at forbinde de to tabeller. Dette betyder, at vi kan få oplysninger om, hvilket land produktet stammer fra.
- *co* er et alias for tabellen *country*. Så for hver ordre vil SQL'en finde det land, hvor produktet stammer fra (via *country_id* i *product*).

- **A single example application/usage of the data set within ONE of the datasets - spotify:**

Your personal domain: *Tempo* kan afsløre, om du foretrækker hurtige eller langsomme sange. Hvis du fx. vil træne, vil du måske prioritere sange med højere *tempo*. For afslappede omgivelser kan du favorisere lavere *tempo*.

A professional domain (You decide whom the "user" is):
Ved at bruge data som *danceability*, kan fx. en DJ objektivt vælge sange, der er mere egnet til at få folk til at danse og skabe den rette stemning ved et event eller i en playliste. I stedet for at vælge sange baseret på intuition eller erfaring alene, kan de bruge *danceability* data for at sikre, at musikken appellerer til publikum og matcher den ønskede atmosfære.

**Research topic to ONE of the datasets:** Formulate a specific question and answer that question with a query

*Hvilke 5 sange udgivet i 2018 har den højeste 'energy', og hvem er 'artist' bag dem?*

```
SELECT artist, song, year,
energy
FROM spotify
WHERE year = 2018
ORDER BY energy DESC
LIMIT 5;
```

| artist | song | year | energy |
|---|---|---|---|
| Sigala | Easy Love | 2018 | 0.9420 |
| Drake | Nice For What | 2018 | 0.9090 |
| Panic! At The… | High Hopes | 2018 | 0.9040 |
| Sofía Reyes | 1, 2, 3 (feat. Jason Derulo & De… | 2018 | 0.8950 |
| Sean Paul | No Lie | 2018 | 0.8820 |