1. Selecting Data

1.1 Querying a database

COUNT()

- Counts the number of records with a value in a field.
- Use an alias for clarity.

```
SELECT COUNT (birthdate) AS count_birthdates
FROM people;
```

count_birthdates
6152

Using * with COUNT()

- `COUNT(field_name)` counts calues in a field.
- `COUNT(*)` counts records in a table.
- `*` represents all fields.

```
SELECT COUNT(*) AS totaql_records
FROM people;
```

total_records
8397

DISTINCT

- `DISTINCT` removes duplicates to return only unique values.

```
SELECT language
FROM films;
```

language
Danish
Danish

language
Greek
Greek
Greek

- Which languages are in our `films` table?

```sql
SELECT DISTINCT language
FROM films;
```

language
Danish
Greek

COUNT() with DISTINCT

- Combine `COUNT()` with `DISTINCT` to count unique values.

```sql
SELECT COUNT(DISTINCT birthdate) AS count_distinct_birthdates
FROM people;
```

count_distinct_birthdates
5398

- `COUNT()` includes duplicates.
- `DISTINCT` excludes duplicates.

1.2 Query execution

Order of execution

- SQL in not processed in its written order

```sql
-- Order of execution
SELECT name -- Secondexecution
FROM people -- First execution
LIMIT 10; -- Third execution
```

- `LIMIT` limits how many rewults we return
- Good to know processing order for debuggign and aliasing

- Aliases are declared in the `SELECT` statement

1.3 SQL style

SQL Formatting

- Formatting is not required

```
select title, release_year, country from filmls limit 3;
```

title	release_year	country
Avengers	2012	USA
Avengers 2	2014	USA
Avengers 3	2018	USA

Best practices

```
SELECT title, release_year, country
FROM filmls
LIMIT 3;
```

title	release_year	country
Avengers	2012	USA
Avengers 2	2014	USA
Avengers 3	2018	USA

Dealing with non-standard field names

- `release year` instead of `release_year`
- Put non-standard field names in double-quotes

```
SELECT title, "release year", country
FROM films
LIMIT 3;
```

2. Filtering Records

2.1 Filtering numbers

Comparison aperators

- `>` Greater than or after
- `<` Less than or before
- `=` Equal to
- `>=` Greater than or equal to
- `<=` Less than or equal to
- `<>` Not equal to

Order of execution

```
-- Written code:
SELECT item
FROM coats
WHERE color = 'green'
LIMIT 5;
```

2.2 Multiple criteria

Multiple criteria

- `OR`, `AND` and `BETWEEN`

```
SELECT *
FROM coats
WHERE color = 'yellow' OR length = 'short';
```

```
SELECT *
FROM coats
WHERE color = 'yellow' AND length = 'short';
```

```
SELECT *
FROM coats
WHERE buttons BETWEEN 1 AND 5;
```

AND, OR

- Silter films released in 1995 or 1995, and certified PF or R
- Enclose individual clauses in parentheses

```
SELECT title
FROM films
WHERE (release_year = 199 OR release_year = 1995)
    AND (certification = 'PG' OR certification = 'R')
```

2.3 Filtering text

- Filter a pattern rather than specific text
- `LIKE`
- `NOT LIKE`
- `IN`

LIKE

Used to search for a pattern in a field

`%` match zero, one or many characters

```
SELECT name
FROM people
WHERE name LIKE 'Ade%';
```

name
Avengers
Avengers 2
Avengers 3

`_` match a singe character

```
SELECT name
FROM people
WHERE name LIKE 'Ev_';
```

name
Eve

Wildcard position

```
SELECT name
FROM people
```

```sql
-- Find results that end on r
WHERE name LIKE '%r';
```

name
A.J Langer
Aaron Seltzer
Aaron Seltzer

```sql
SELECT name
FROM people
-- Find results which the third character is t
WHERE name LIKE '__t%';
```

name
Aitana
Anthony

WHERE, In

```sql
SELECT title
FROM films
WHERE release_year IN (1920, 1930, 1940);
```

title
Over the Hill
Hell's angels

2.4 NULL values

Missing values

- `COUNT(field_name)` includes only mom-missing values
- `COUNT(*)` includes missing values

`NULL`

- Missing values:
- Human error
- Information not available
- Unknown

IS NULL

```sql
SELECT name
FROM people
WHERE birthdate IS NULL;
```

name
A. Raven Cruz
A.J. DeLucia