

# SQL queries – filtering & aggregate functions

- Filtering (WHERE)
- Aggregate functions

# SQL query

- A bit more complex example:

```
SELECT <fieldlist>  
FROM <tablename>  
WHERE <condition>
```

Which fields are wanted?

From which table should the fields be taken?

Which condition should apply?

# SQL query

- The WHERE part is a logical expression, specifying conditions on certain fields
- Five fundamental types of criteria
  - Comparison (<, >, =)
  - Range (< AND >)
  - Set membership (belongs to a set of values)
  - Pattern match (for string fields)
  - Null (is the value of the field a null value)

# Tabel: Guest

Guest_No	Name	Address
1	Eva	Paradisvej 3, 1111 Bispeborg
2	Adam	Paradisvej 7, 1111 Bispeborg
3	Goeg	Sunset Blvd. 8, 2222 Hjemby
4	Gokke	Sunset Blvd. 8, 2222 Hjemby
5	Fy	Klovnevej 87, 3333 Lilleby
6	Bi	Paradisvej 100, 1111 Bispeborg
7	Romeo	Paradisvej 100, 1111 Bispeborg
8	Julie	Paradisvej 100, 1111 Bispeborg
9	Godzilla	Paradisvej 100, 1111 Bispeborg
10	KingKong	Paradisvej 100, 1111 Bispeborg

```
SELECT *  
FROM Guest  
WHERE Guest_No < 5
```

# SQL query

- You can build compound conditions using ***logical expressions*** by the use of the logical operators: **AND, OR, NOT**
- Rules are the same as for logical expressions in C# (or other languages)
- Use parenthesis to increase readability and/or to "overrule" rules

# Tabel: Guest

Guest_No	Name	Address
1	Eva	Paradisvej 3, 1111 Bispeborg
2	Adam	Paradisvej 7, 1111 Bispeborg
3	Goeg	Sunset Blvd. 8, 2222 Hjemby
4	Gokke	Sunset Blvd. 8, 2222 Hjemby
5	Fy	Klovnevej 87, 3333 Lilleby
6	Bi	
7	Romeo	
8	Julie	
9	Godzilla	
10	KingKong	

```
SELECT Name  
FROM Guest  
WHERE Guest_No < 5
```

# SQL query - interval

- A **range search** is an SQL query where a value should be within a certain range
- Actually just a two-part comparison query

```
SELECT *  
FROM Guest  
WHERE ((Guest_no <= 6) AND (Guest_no >= 3))
```

# SQL query - interval

- Another notation for range search uses the keyword **BETWEEN**

```
SELECT *
```

```
FROM Guest
```

```
WHERE Guest_no BETWEEN 1 AND 6
```



# SQL query - interval

- We can create a "negated" version of a range query using **NOT BETWEEN**

```
SELECT *  
FROM Guest  
WHERE Guest_no NOT BETWEEN 1 AND 6
```

# Exercise – SQL queries

- Formulate queries to get the below data:
  - Get all fields for rooms where the type is 'F' in the hotel with Hotel\_no = 1
  - Get all fields for rooms that are not a 'F' family or a 'D' double room
  - Get all bookings that are after the 15.3.2011
  - Get all bookings that are after the 15.3.2011 but also before the 15.4.2011
  - Get all bookings for hotel\_no = 1 and guest\_no = 2 that are after the 15.3.2011 but also before the 15.4.2011
  - Get all booking for Hotel\_no = 2
- Formulate your own queries.

# SQL query – set membership

- A **set membership search** is an SQL query where a value must belong to a given set of values
- We use the **IN** keyword

```
SELECT *  
FROM Guest  
WHERE Name IN ('Adam','Eva')
```

# SQL query – set membership

- Note that these two queries are equivalent

```
SELECT *  
FROM Guest  
WHERE Name IN ('Adam','Eva')
```

```
SELECT *  
FROM Guest  
WHERE ((Name = 'Adam') OR (Name = 'Eva'))
```

# SQL query – set membership

- We can create a "negated" version of a set membership query using **NOT IN**

```
SELECT *  
FROM Guest  
WHERE Name NOT IN ('Adam','Eva')
```

# Exercise – SQL queries

- Now formulate queries yourself, in order to retrieve the below data:
  - Get all guests from booking where hotel\_no 1, 3, 4
  - Get all rooms from room where hotel\_no = 1 that are not a double or family room
  - Get all guest that have a booking, but not in the period 15.3.2011 to 15.4.2011
- Formulate your own queries

# SQL query – mønster søgning (eng.: pattern match)

- A **pattern match search** is an SQL query where a (string) value must match a given pattern
- We use the **LIKE** keyword
- The hard part is choosing the correct pattern to match against – several ways to formulate a pattern

# SQL query – pattern match

- A pattern is formulated using two special characters % and \_
- % : wildcard: any sequence of zero or more characters
- \_ : any single character





# SQL query – pattern match

Pattern	Meaning
'S%'	Any string starting with 'S', of any length (at least 1) ( 'super', 's', 's123', 's 123' )
'S____'	Any string starting with 'S', of length exactly 4 ( 'such', 's123', 'ssss', 's 1' )
'%S'	Any string ending with 's', of any length (at least 1) ( 'Spurs', 's', '123s', ' s', '1 2s' )
'%S%'	Any string containing an 's', of any length (at least 1) ( 'Spurs', 's', 'basin', ' s ', '12s34' )
'%S____%'	Exercise...

# SQL query – pattern match

```
SELECT *  
FROM Guest  
WHERE Name LIKE 'P%'
```

```
SELECT *  
FROM Guest  
WHERE Name LIKE '____'
```

# SQL query – pattern match

- We can create a "negated" version of a pattern match query using **NOT LIKE**

```
SELECT *  
FROM Hotel  
WHERE Name NOT LIKE 'D%'
```

# SQL query – null

- A **null search** is a query, where the value must be null
- We use the keyword **IS NULL**
- You can allow a field to have an undefined or null value, if it makes sense

```
SELECT *  
FROM Guest  
WHERE Address IS NULL
```

# SQL query – null (negated)

- We can make a "negated" version of a null query by using **IS NOT NULL**

```
SELECT *
```

```
FROM Guest
```

```
WHERE Address IS NOT NULL
```

# Exercise – SQL queries

- With the data in place, then run the below queries
  - `SELECT * FROM Hotel WHERE name LIKE '%D%'`
  - `SELECT * FROM hotel WHERE Address LIKE '%n'`
  - `SELECT * FROM Hotel WHERE Address LIKE '%_____ %'`
  - `SELECT * FROM Booking WHERE Date_From IS NOT NULL`
- Formulate queries to get the below data:
  - Get all hotels from Roskilde
  - Get all hotels
  - Get bookings with a value for Date\_from but no value for Date\_to
  - Get all hotels with a name starting with 'P' and a length of 4 characters
  - Get all hotels, where the name contains a 'P' or a 'p'
- Formulate your own queries

# SQL query - functions

- Simple arithmetic can be executed in SQL by using the below functions
  - COUNT
  - SUM
  - AVG
  - MIN
  - MAX
- These functions are called **aggregate functions**

# SQL query - functions

- An aggregate function work on values for a specific field (column)
- The condition determines, which records to be aggregated

Example: how many rooms are there in Hotel no. 1?

```
SELECT count(*) FROM room WHERE  
Hotel_No = 1;
```



# SQL query - functions

- This query can also be written as

```
SELECT count(*) AS Number_of_Rooms  
FROM room WHERE Hotel_No = 1;
```

- The key word **AS** gives the possibility to rename a field in the search result
- Only cosmetic, but very useful...

# SQL query - functions

```
SELECT
    COUNT(*) AS Number_of_Rooms,
    AVG(Price) as Avg_price,
    Max(Price)
FROM room
where Hotel_no = 1;
```

# Exercise – SQL queries

- With the data in place for the *hotel database* then run the below queries:
  - `SELECT * FROM Booking ORDER BY Hotel_no ASC, Room_NO DESC`
  - `SELECT Name, Address FROM Hotel ORDER BY Hotel_No;`
  - `SELECT MAX(Price) AS maxPrice FROM room WHERE Types = 'D';`
- Formulate yourself queries to get the below information:
  - Get all bookings from hotel no. 2 (oldest first)
  - Get a sorted list of name and address of guests