5602201

2. SQL Basics

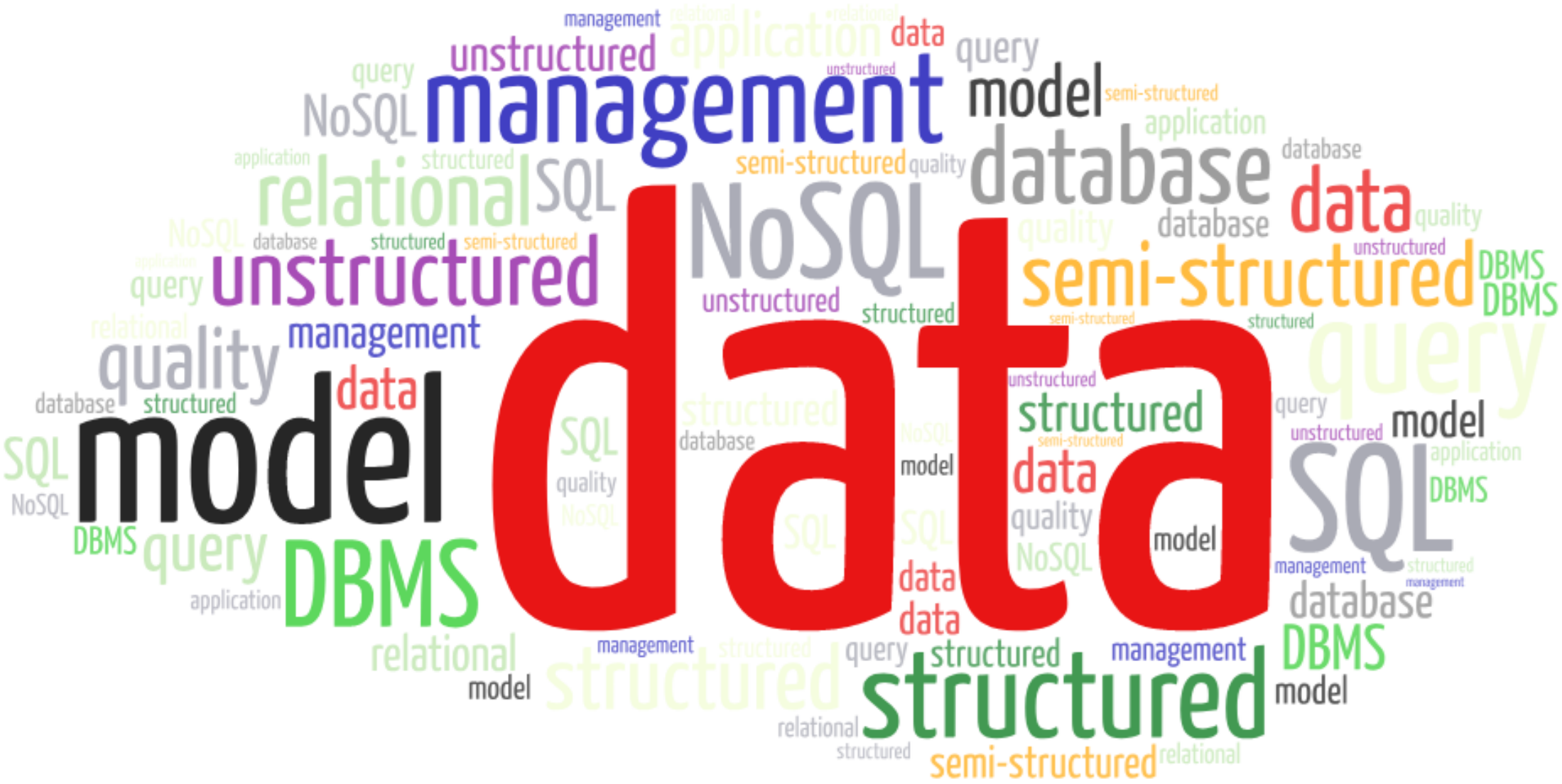
CHUTIPORN ANUTARIYA


LET'S
START

WARM UP Discussion:

Data …

What are you thinking of when we talk about data?


data
management
unstructured
application
data
query
model
semi-structured
NoSQL
relational
SQL
database
quality
structured
DBMS

Let's Discuss:

Sources of Data …

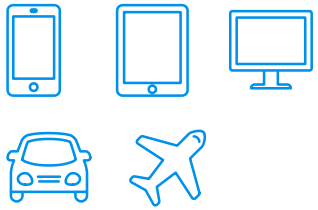
Where do data come from?

Who/what generates data?

Three major sources of data...

Machines

Data generated from real time sensors, machines, vehicles, web logs, etc.



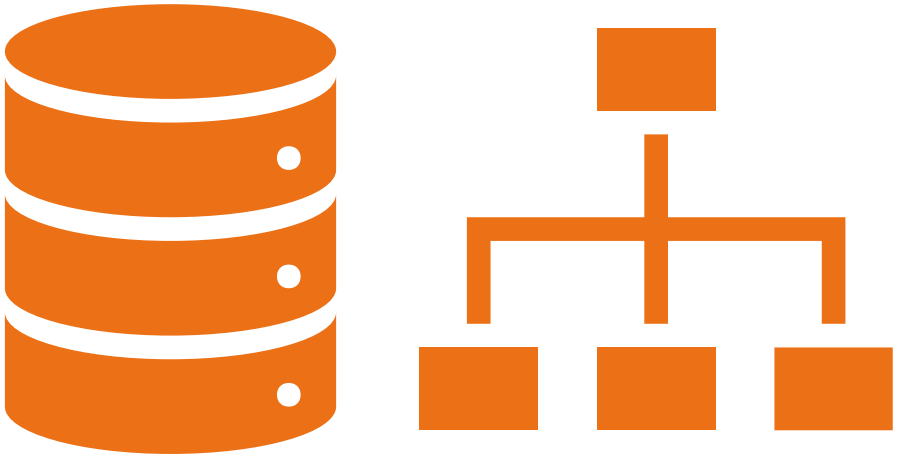
People

Tweets, status updates, social media data, photos, videos

Organizations

More traditional type of data: transaction information, databases, data warehouses

RELATIONAL MODEL CONCEPTS



RELATIONAL MODEL STRUCTURE

The Relational Data Model

Relational model

- First commercial implementations available in early 1980s
- Has been implemented in a large number of commercial system

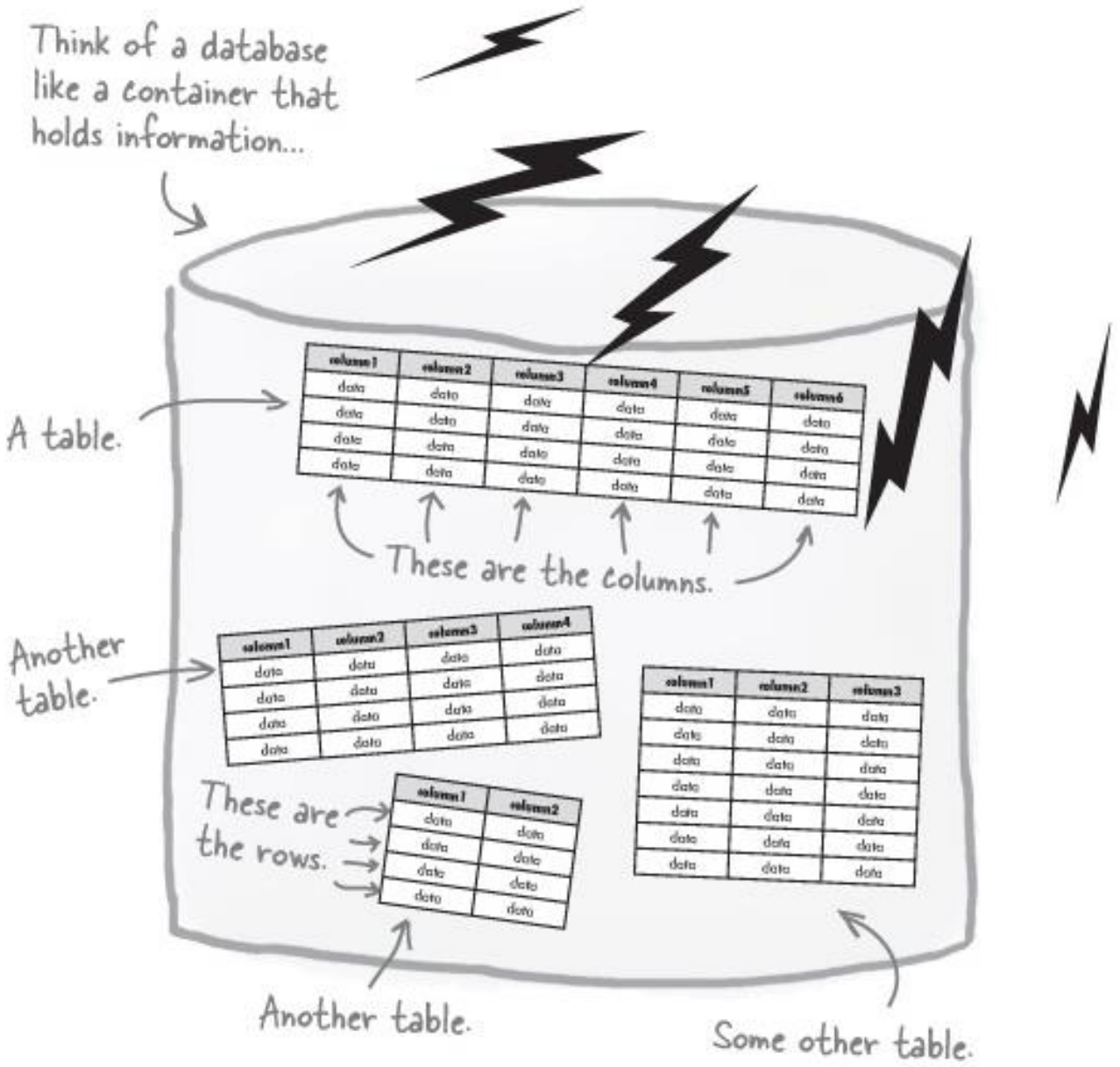
Hierarchical and network models

- Preceded the relational model

Anatomy of a Database

Database = collection of tables

The information inside the database is organized into tables.



Relational Model Concepts

Table Name

Columns / attributes / properties

Rows / tuples / records

STUDENT

Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Benjamin Bayer	305-61-2435	(817)373-1616	2918 Bluebonnet Lane	NULL	19	3.21
Chung-cha Kim	381-62-1245	(817)375-4409	125 Kirby Road	NULL	18	2.89
Dick Davidson	422-11-2320	NULL	3452 Elgin Road	(817)749-1253	25	3.53
Rohan Panchal	489-22-1100	(817)376-9821	265 Lark Lane	(817)749-6492	28	3.93
Barbara Benson	533-69-1238	(817)839-8461	7384 Fontana Lane	NULL	19	3.25

Relation Schema

Relation schema R

- Denoted by R(A1, A2, ...,An)
- Made up of a relation name R and a list of attributes, A1, A2, ..., An

Attribute Ai

- Name of a role played by some domain D in the relation schema R

Example: Relation Schema STUDENT

A relation which stores information about university students, would contain seven attributes describing each student:

- STUDENT(Name, Ssn, Home_phone, Address, Office_phone, Age, Gpa)

or (with the data type of each attribute specified)

- STUDENT(Name: string, Ssn: string, Home_phone: string, Address: string, Office_phone: string, Age: integer, Gpa: real)

Characteristics of Tables: Order of rows

The order of rows in a table is not important.

STUDENT

Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Dick Davidson	422-11-2320	NULL	3452 Elgin Road	(817)749-1253	25	3.53
Barbara Benson	533-69-1238	(817)839-8461	7384 Fontana Lane	NULL	19	3.25
Rohan Panchal	489-22-1100	(817)376-9821	265 Lark Lane	(817)749-6492	28	3.93
Chung-cha Kim	381-62-1245	(817)375-4409	125 Kirby Road	NULL	18	2.89
Benjamin Bayer	305-61-2435	(817)373-1616	2918 Bluebonnet Lane	NULL	19	3.21

Characteristics of Tables: Values

Each value in a row is atomic

Flat relational model

- Composite and multivalued attributes not allowed
- First normal form assumption

Multivalued attributes

- Must be represented by separate relations

Composite attributes

- Represented only by simple component attributes in basic relational model

Characteristics of Tables: NULL values

Represent the values of attributes that may be unknown or may not apply to a row

Meanings for NULL values

- *Value unknown*
- *Value exists but is not available*
- *Attribute does not apply to this tuple (aka. value undefined)*

Characteristics of Tables: Meaning

Interpretation (meaning) of a table

- Each row in the table is a **fact** or a particular instance of the assertion

RELATIONAL MODEL CONSTRAINTS

Relational Model Constraints

Constraints

- Restrictions on the actual values in a database state
- Derived from the rules in the application that the database represents

Relational Model Constraint Categories

Inherent model-based constraints or implicit constraints

- Inherent in the data model
- The characteristics of relations discussed earlier belong to this category
- Ex: The constraint that a relation cannot have duplicate tuples

Schema-based constraints or explicit constraints

- Can be directly expressed in schemas of the data model
- Ex: Domain constraints, Key constraints, NULL value constraints, etc. (to be discussed next)

Application-based or semantic constraints or business rules

- Cannot be directly expressed in schemas
- Expressed and enforced by application program

Domain and Data Type Constraints

Domain constraints

- Specify that within each row, the value of each column A must be an atomic value from the domain dom(A), such as:
  - Numeric data types for integers and real numbers
  - Characters
  - Booleans
  - Fixed-length strings
  - Variable-length strings
  - Date, time, timestamp
  - Money
  - Other special data types

Key Constraints

Two distinct rows in a table cannot have identical values for (all) attributes in key

CAR

License_number	Engine_serial_number	Make	Model	Year
Texas ABC-739	A69352	Ford	Mustang	02
Florida TVP-347	B43696	Oldsmobile	Cutlass	05
New York MPO-22	X83554	Oldsmobile	Delta	01
California 432-TFY	C43742	Mercedes	190-D	99
California RSK-629	Y82935	Toyota	Camry	04
Texas RSK-629	U028365	Jaguar	XJS	04

Figure 3.4
The CAR relation, with two candidate keys: License_number and Engine_serial_number.

Key Constraints (cont'd.)

Candidate key

- Relation schema may have more than one key

Primary key of the relation

- Designated among candidate keys
- Underline attribute

Other candidate keys are designated as **unique keys**

Primary Key Constraints

The primary key is used to uniquely identify each record

Which means that the data in the primary key column can't be repeated.

Consider a table with the columns shown below. Do you think any of those would make good primary keys?

SSN	last_name	first_name	phone_number

NULL Value Constraint

A NULL value constraint specifies whether NULL values are or are not permitted.

Example

- If every STUDENT row must have a valid, not-NULL value for the Name attribute, then Name of STUDENT is constrained to be NOT NULL.

Referential Integrity / Foreign Key Constraint

Referential integrity (or Foreign Key) constraint

- Connecting two tables
- Specified between two tables
- Maintains consistency among rows in two tables
- The FOREIGN KEY (FK) is a column in a table that references the PRIMARY KEY of another table.

SQL: Basics

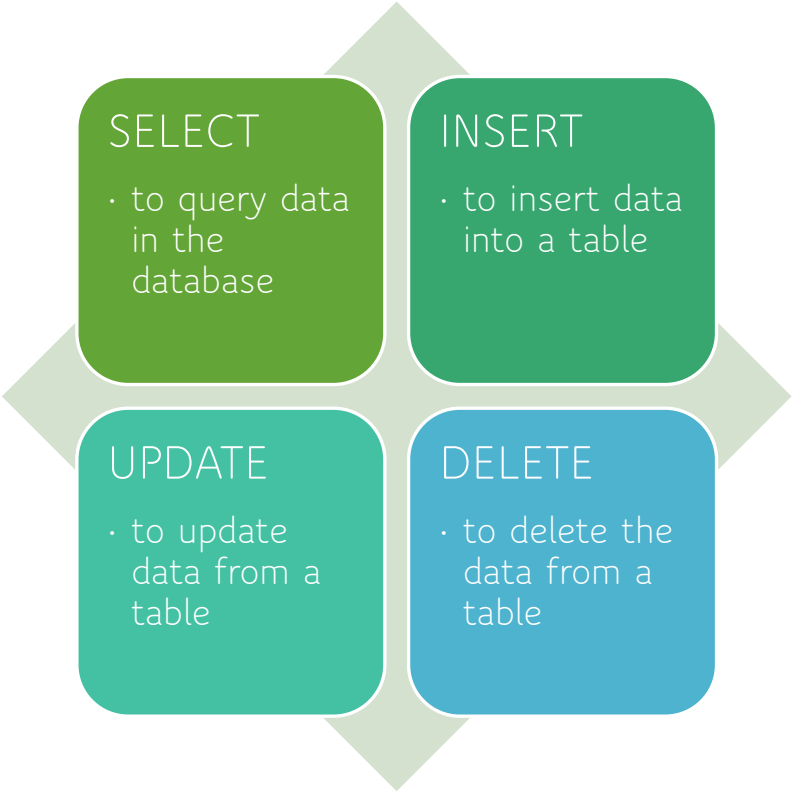
STRUCTURED QUERY LANGUAGE

SQL: Structured Query Language

- SQL became a standard of the American National Standards Institute (ANSI) in 1986, and of the International Organization for Standardization (ISO) in 1987.
- SQL standard has two main components:
  - Data Definition Language (DDL): to define the structure of the database and control data access.
  - Data Manipulation Language (DML): for data retrieval and updating.

Data Manipulation Language (DML)

Data Manipulation Language (DML): for data retrieval and updating.



Demo Database and SQL Playground

https://www.w3schools.com/sql/trysql.asp?filename=trysql_select_all

Your Database:

Tablename	Records
Customers	91
Categories	8
Employees	10
OrderDetails	518
Orders	196
Products	77
Shippers	3
Suppliers	29

SQL SELECT Statement

The SELECT statement is used to select data from a database.

SELECT Syntax

```sql
SELECT column1, column2, ...
FROM table_name;
```

SELECT

https://www.w3schools.com/sql/trysql.asp?filename=trysql_select_all

Example

```
SELECT * FROM Customers;
```

Example

```
SELECT Country FROM Customers;
```

SQL SELECT DISTINCT Statement

The SELECT DISTINCT statement is used to return only distinct (different) values.

Inside a table, a column often contains many duplicate values; and sometimes you only want to list the different (distinct) values.

SELECT DISTINCT Syntax

```sql
SELECT DISTINCT column1, column2, ...
FROM table_name;
```

SELECT DISTINCT Examples

The following SQL statement selects only the DISTINCT values from the "Country" column in the "Customers" table:

Example

```
SELECT DISTINCT Country FROM Customers;
```

The following SQL statement lists the number of different (distinct) customer countries:

Example

```
SELECT COUNT(DISTINCT Country) FROM Customers;
```

SQL WHERE Clause

The WHERE clause is used to filter records.

It is used to extract only those records that fulfill a specified condition.

WHERE Syntax

```
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

Example

```
SELECT * FROM Customers
WHERE Country='Mexico';
```

Example

```
SELECT * FROM Customers
WHERE CustomerID=1;
```

SQL WHERE Clause

Operators in The WHERE Clause

Operator	Description
=	Equal
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal
<>	Not equal. **Note:** In some versions of SQL this operator may be written as !=
BETWEEN	Between a certain range
LIKE	Search for a pattern
IN	To specify multiple possible values for a column

SQL Statement:

```
SELECT * FROM Products
WHERE Price >= 30;
```

SQL Statement:

```
SELECT * FROM Customers
WHERE City LIKE 's%';
```

SQL Statement:

```
SELECT * FROM Products
WHERE Price <> 18;
```

Example

```
SELECT * FROM Customers
WHERE City LIKE '_ondon';
```

SQL Statement:

```
SELECT * FROM Products
WHERE Price BETWEEN 50 AND 60;
```

SQL Statement:

```
SELECT * FROM Customers
WHERE City IN ('Paris','London');
```

SQL WHERE Clause

AND Syntax

```
SELECT column1, column2, ...
FROM table_name
WHERE condition1 AND condition2 AND condition3 ...;
```

OR Syntax

```
SELECT column1, column2, ...
FROM table_name
WHERE condition1 OR condition2 OR condition3 ...;
```

NOT Syntax

```
SELECT column1, column2, ...
FROM table_name
WHERE NOT condition;
```

SQL AND, OR and NOT Operators

Example

```
SELECT * FROM Customers
WHERE Country='Germany' AND City='Berlin';
```

Example

```
SELECT * FROM Customers
WHERE City='Berlin' OR City='München';
```

Example

```
SELECT * FROM Customers
WHERE NOT Country='Germany';
```

SQL AND, OR and NOT Operators

1 List names of customers who are from Germany and the city must be Berlin OR Aachen (use parenthesis to form complex expressions).

1.1 Write SQL statement

1.2 What are the answers?

2 List names of customers who are NOT from Norway and from Denmark.

2.1 Write SQL stement

2.2 How many records are the answers?

SQL ORDER BY Keyword

The ORDER BY keyword is used to sort the result-set in ascending or descending order.

The ORDER BY keyword sorts the records in ascending order by default. To sort the records in descending order, use the DESC keyword.

```
SELECT column1, column2, ...
FROM table_name
ORDER BY column1, column2, ... ASC|DESC;
```

Example

```
SELECT * FROM Customers
ORDER BY Country;
```

Example

```
SELECT * FROM Customers
ORDER BY Country DESC;
```

Example

```
SELECT * FROM Customers
ORDER BY Country, CustomerName;
```

SQL ORDER BY Keyword

3

Selects all customers, sorted ascending by Country and descending by the CustomerName.

3.1 Write SQL statement

3.2 Who is listed as the first customer. Give the customer name.

Questions?