Databases

Relational Databases and MySQL

SoftUni Team

Technical Trainers

Software University

https://about.softuni.bg

Table of Contents

Have a Question?

Software University

sli.do

#qa-fund

Relational Databases

Tables, Relationships and SQL

What is a Database?

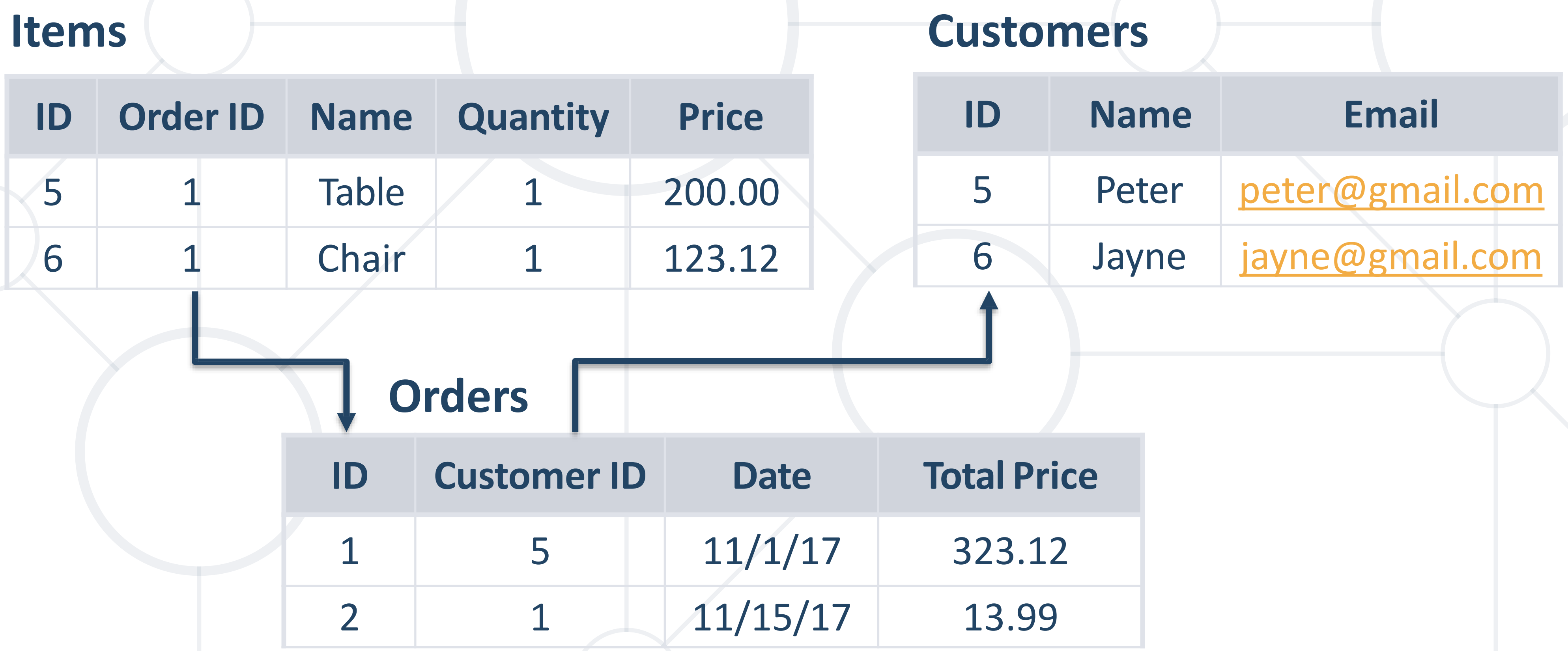
- A **database** is a collection of data, organized to be easily accessed, managed and updated
- Modern databases are managed by **Database Management Systems** (DBMS)
  - Define database **structure**, e. g. tables, collections, columns, relations, indexes
  - **C**reate / **R**ead / **U**pdate / **D**elete data (**CRUD** operations)
  - Execute **queries** (filter / search data)

SQL Databases (Relational Databases)

- Relational (**SQL**) databases organize data in **tables**
  - Tables have strict structure (**columns** of certain **data types**)
  - Can have **relationships** to other tables
- Relational databases use the **structured query language** (**SQL**) for defining and manipulating data
  - Extremely powerful for complex queries
- **Relational databases** are the most widely used data management technology

The Relational DB Model

- Relational data is stored into one or more **tables** with a **unique key** identifying each row and **foreign keys** defining **relationships**



Items

ID	Order ID	Name	Quantity	Price
5	1	Table	1	200.00
6	1	Chair	1	123.12

Customers

ID	Name	Email
5	Peter	peter@gmail.com
6	Jayne	jayne@gmail.com

Orders

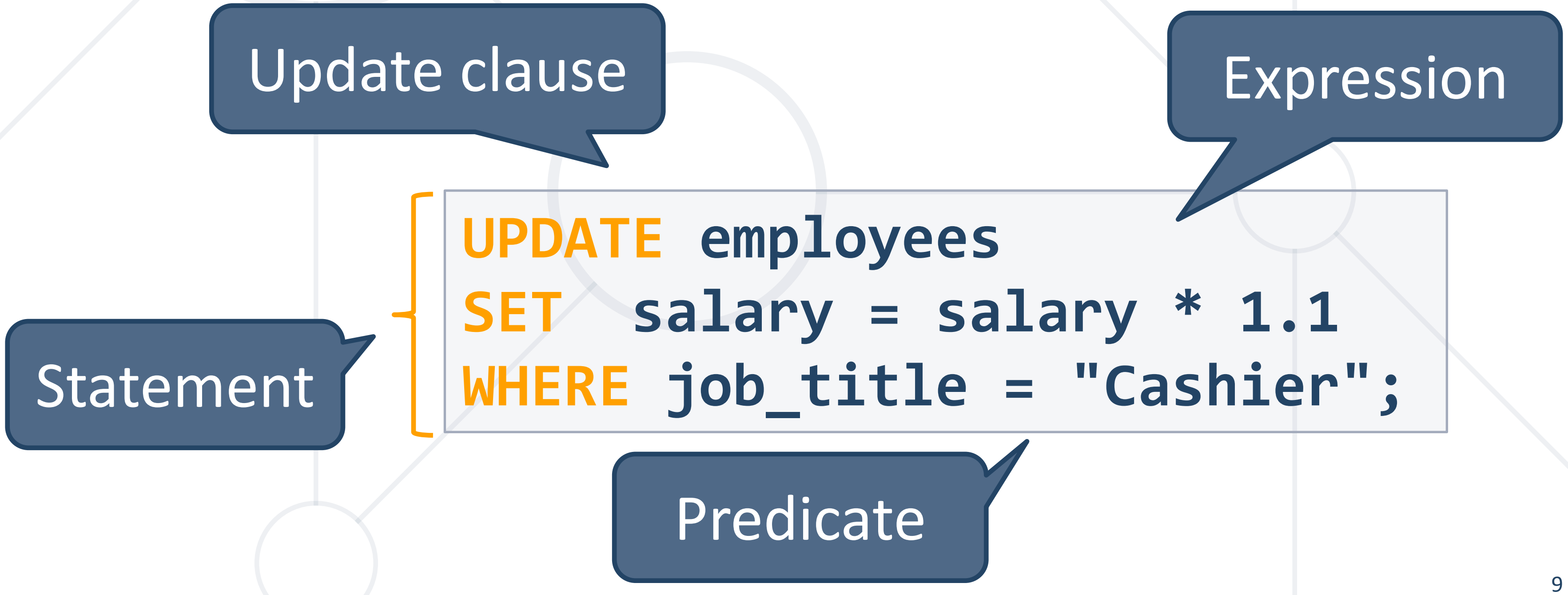
ID	Customer ID	Date	Total Price
1	5	11/1/17	323.12
2	1	11/15/17	13.99

SQL and MySQL

Powerful Data Management

Structured Query Language (SQL)

- **SQL** == query language designed for managing data in **relational** databases (RDBMS)
- Subdivided into several language elements
  - Queries
  - Clauses
  - Expressions
  - Predicates
  - Statements



```sql
UPDATE employees
SET   salary = salary * 1.1
WHERE job_title = "Cashier";
```

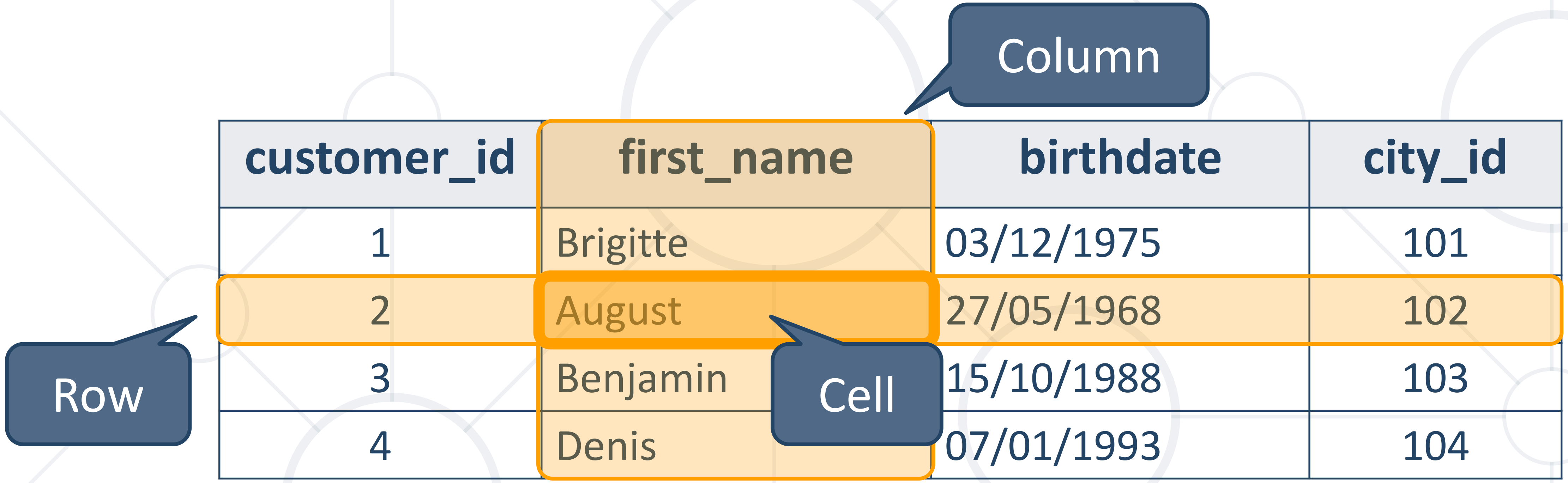
Structured Query Language (2)

- Logically divided in four sections:
  - **Data Definition (DDL)** – describe the structure of our data
  - **Data Manipulation (DML)** – store and retrieve data
  - **Data Control (DCL)** – define who can access the data
  - **Transaction Control (TCL)** – bundle operations and allow rollback

DDL	DML	DCL	TCL
CREATE	SELECT	GRANT	BEGIN TRAN
ALTER	INSERT	REVOKE	COMMIT
DROP	UPDATE	DENY	ROLLBACK
TRUNCATE	DELETE		SAVE

Database Table Elements

- The table is the main **building block** of any database



customer_id	first_name	birthdate	city_id
1	Brigitte	03/12/1975	101
2	August	27/05/1968	102
3	Benjamin	15/10/1988	103
4	Denis	07/01/1993	104

- Each **row** is called a **record** or **entity**
- Columns (**fields**) define the **type** of data they contain

Why MySQL?

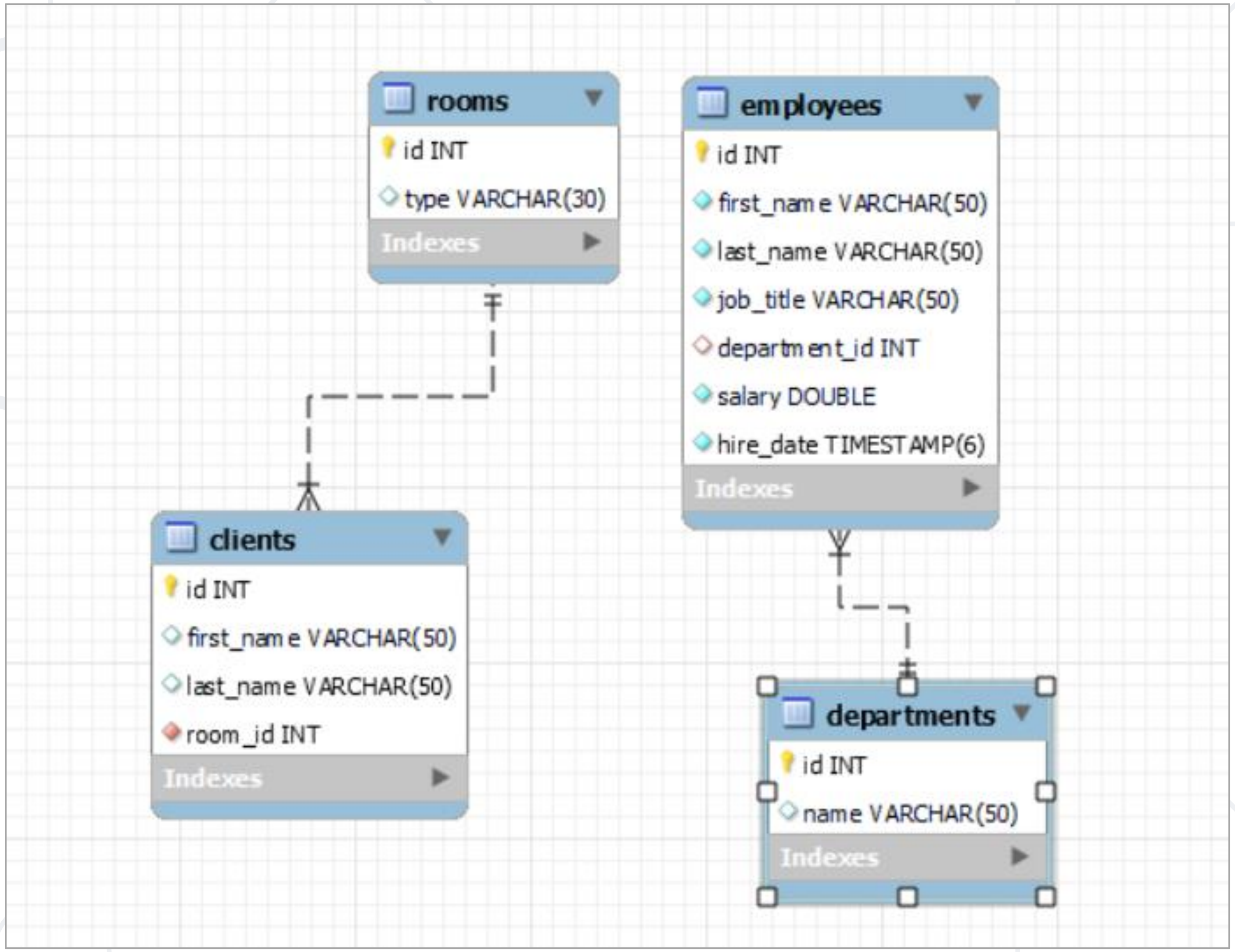
- MySQL is a specific database management system (**DBMS**)
- It is a software that **implements the SQL language** and provides a platform to store, manage, and retrieve data efficiently
- One of the **most popular** and **widely used** relational database management systems
- **Open-source** and **free** to use
- Download **MySQL Community Server**
  - **Windows**: `https://dev.mysql.com/downloads/mysql/`
  - **Ubuntu/Debian**: `https://dev.mysql.com/downloads/repo/apt/`

Retrieving Data

Using SQL SELECT

Hotel Database

- Run the Hotel_DB.sql script to create the database

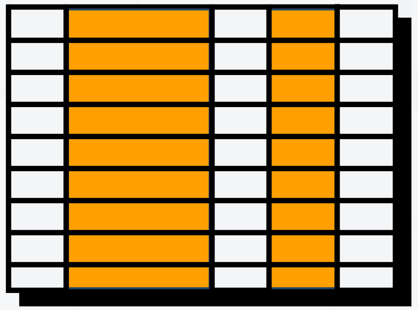


Capabilities of SQL SELECT

Software University

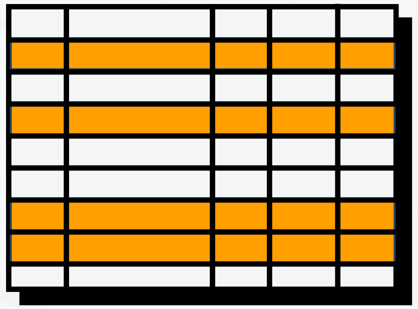
Projection

Take a subset of the columns



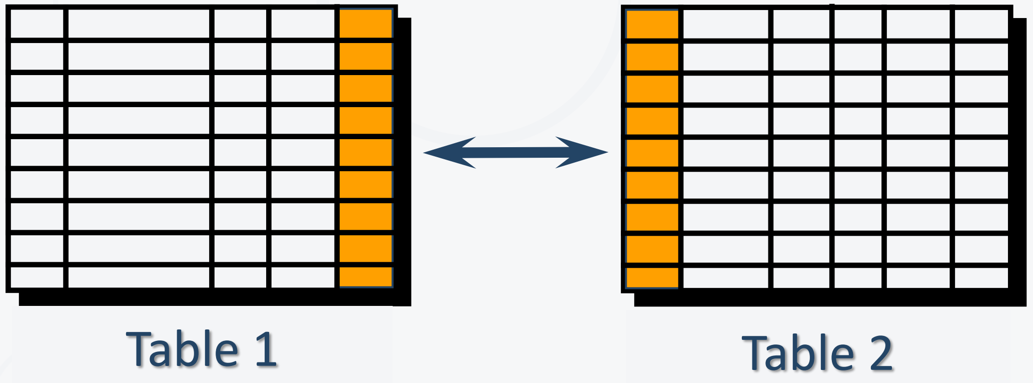
Selection

Take a subset of the rows



Join

Combine tables by some column



Related Tables

- We split the data and introduce **relationships** between the tables to **avoid** repeating information

user_id	first	last	registered
203	David	Rivers	05/02/2016
204	Sarah	Thorne	07/17/2016
205	Michael	Walters	11/23/2015

user_id	email
203	drivers@mail.cx
204	sarah@mail.cx
205	walters_michael@mail.cx
203	david@homedomain.cx

Primary Key

Foreign Key

- Connection via **Foreign Key** in one table pointing to the **Primary Key** in another

SELECT – Examples

- Selecting all columns from the "employees" table

id	first_name	last_name	job_title	department_id	salary
1	John	Smith	Manager	1	900
2	John	Johnson	Customer Service	1	880
3	Smith	Johnson	Porter	2	1100
…	…	…	…	…	…

```sql
SELECT * FROM employees;
```

List of columns (* for all)

Table name

Problem: Select Employee Information

- Write a query to **select** all employees from "**Hotel**" database
  - **Retrieve** information about their **id, first_name, last_name** and **job_title**
  - **Ordered** by id
- Note: Query **Hotel** database

id	first_name	last_name	job_title
1	John	Smith	Manager
2	John	Johnson	Customer Service
3	Smith	Johnson	Porter
…	…	…	…

Solution: Select Employee Information

```
SELECT id, first_name, last_name, job_title
FROM employees
ORDER BY id;
```

List of columns

Table name

- **Aliases** rename a table or a column heading:

```
SELECT e.id AS 'No.',
e.first_name AS 'First Name',
e.last_name AS 'Last Name',
e.job_title AS 'Job Title'
FROM employees AS e ORDER BY id;
```

Concatenation

- **concat()** - returns the string that results from concatenating the arguments
  - String literals are enclosed in ['](**single quotes**)
  - Table and column names containing special symbols use [`] (**backtick**)

```
SELECT concat(`first_name`,' ',`last_name`) AS 'Full Name',
     `job_title` as  'Job Title',
       `id` AS 'No.'
  FROM `employees`;
```

Problem: Select Employees with Filter

- Find information about all employees, listing their:
  - **Full Name**
  - **Job title**
  - **Salary**
- Use **concatenation** to display first and last names as **one field**
- Note: Query **Hotel** database

Solution: Select Employees with Filter

Concatenation

```
SELECT concat(`first_name`,' ',`last_name`) AS
     'Full name',
     `job_title` AS 'Job title',
       `salary` AS 'Salary'
  FROM `employees` WHERE salary > 1000;
```

Column alias

Filtering the Selected Rows

- Use **DISTINCT** to eliminate duplicate results

```
SELECT DISTINCT department_id
FROM employees;
```

- You can filter rows by specific conditions using the **WHERE** clause

```
SELECT last_name, department_id
FROM employees
WHERE department_id = 1;
```

- Other **logical operators** can be used for better control

```
SELECT last_name, salary
FROM employees
WHERE salary <= 2000;
```

Other Comparison Conditions

- Conditions can be combined using **NOT**, **OR**, **AND** and brackets

```
SELECT first_name, last_name FROM employees
WHERE NOT (department_id = 3 OR department_id = 4);
```

- Using **BETWEEN** operator to specify a range:

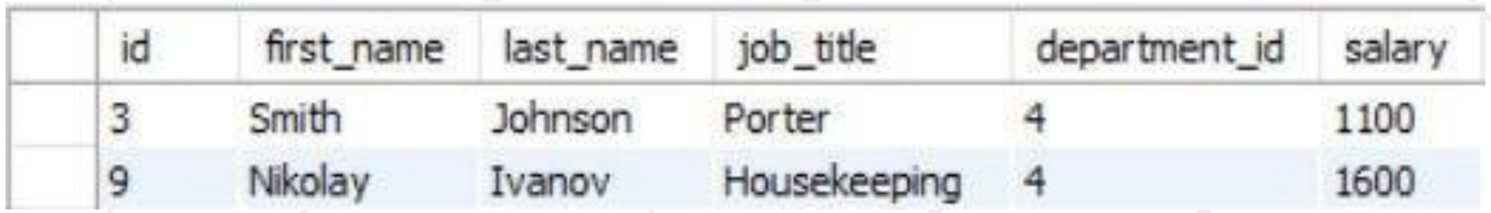
```
SELECT last_name, salary FROM employees
WHERE salary BETWEEN 1200 AND 2200;
```

- Using **IN / NOT IN** to specify a set of values:

```
SELECT first_name, last_name, department_id
FROM employees
WHERE department_id IN (1,3,4);
```

Problem: Select Employees by Multiple Filters

- Write a query to **retrieve** information about employees, order by id
  - Who are in **department 4**
  - Have salary **higher or equal to 1000**



id	first_name	last_name	job_title	department_id	salary
3	Smith	Johnson	Porter	4	1100
9	Nikolay	Ivanov	Housekeeping	4	1600

```sql
SELECT * FROM employees AS e
WHERE e.department_id = 4 AND e.salary >= 1000;
```

Comparing with NULL

- NULL is a special value that means missing value
  - Not the same as 0 or a blank space
- Checking for NULL values

This is false and it won't return results!

```sql
SELECT last_name, department_id
FROM employees
WHERE department_id = NULL;
```

```sql
SELECT last_name, department_id
FROM employees
WHERE department_id IS NULL;
```

```sql
SELECT last_name, department_id
FROM employees
WHERE department_id IS NOT NULL;
```

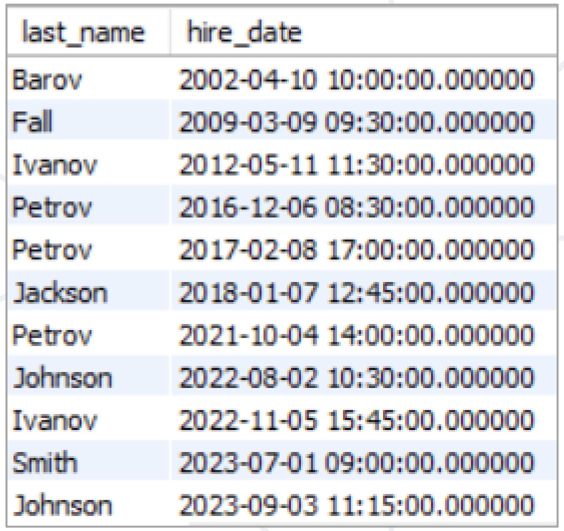
Sorting with ORDER BY

- Sort rows with the **ORDER BY** clause
  - **ASC**: ascending order, default

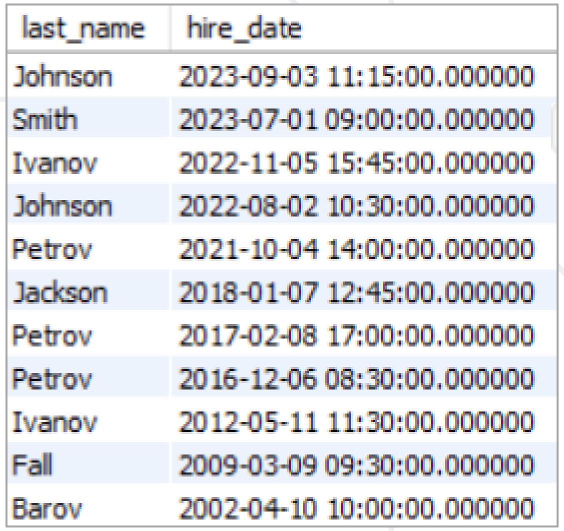
```
SELECT last_name, hire_date
FROM employees
ORDER BY hire_date;
```

  - **DESC**: descending order

```
SELECT last_name, hire_date
FROM employees
ORDER BY hire_date DESC;
```



last_name	hire_date
Barov	2002-04-10 10:00:00.000000
Fall	2009-03-09 09:30:00.000000
Ivanov	2012-05-11 11:30:00.000000
Petrov	2016-12-06 08:30:00.000000
Petrov	2017-02-08 17:00:00.000000
Jackson	2018-01-07 12:45:00.000000
Petrov	2021-10-04 14:00:00.000000
Johnson	2022-08-02 10:30:00.000000
Ivanov	2022-11-05 15:45:00.000000
Smith	2023-07-01 09:00:00.000000
Johnson	2023-09-03 11:15:00.000000



last_name	hire_date
Johnson	2023-09-03 11:15:00.000000
Smith	2023-07-01 09:00:00.000000
Ivanov	2022-11-05 15:45:00.000000
Johnson	2022-08-02 10:30:00.000000
Petrov	2021-10-04 14:00:00.000000
Jackson	2018-01-07 12:45:00.000000
Petrov	2017-02-08 17:00:00.000000
Petrov	2016-12-06 08:30:00.000000
Ivanov	2012-05-11 11:30:00.000000
Fall	2009-03-09 09:30:00.000000
Barov	2002-04-10 10:00:00.000000

Writing Data in Tables

Using SQL INSERT

Inserting Data (1)

- The SQL **INSERT** command

```
INSERT INTO departments (name) VALUES ('Human Resources');
```

```
INSERT INTO employees (first_name, last_name, job_title,
department_id, salary, hire_date)
VALUES
('Michael', 'Scott', 'Regional Manager', 5, 2500.00,
'2023-07-12 09:30:00');
```

Specify columns

- **Bulk data** can be recorded in a single query, separated by comma

Inserting Data (2)

- You can use existing records to create a **new table**

```
CREATE TABLE employee_contacts
AS
SELECT id, first_name, CONCAT(first_name, '.', last_name,
'@hotel.com’)
AS email,
'N/A' AS phone
FROM employees;
```

New table name

Existing source

Modifying Existing Records

Using SQL UPDATE and DELETE

Updating Data

- The SQL **UPDATE** command

```sql
UPDATE employees
   SET last_name = 'Brown'
 WHERE employee_id = 1;
```

New values

```sql
UPDATE employees
   SET salary = salary * 1.10,
       job_title = CONCAT('Senior',' ', `job_title`)
 WHERE department_id = 3;
```

- Note: Don't forget the **WHERE** clause!

Problem: Update Employees Salary

- **Update** all employees salaries whose **job_title** is "**Housekeeper**" by **100**

```sql
UPDATE employees
SET salary = salary + 100
WHERE job_title = 'Housekeeper';
SELECT salary
FROM employees;
```

Deleting Data

- Deleting specific rows from a table

```
DELETE FROM employees
WHERE employee_id = 1;
```

Condition

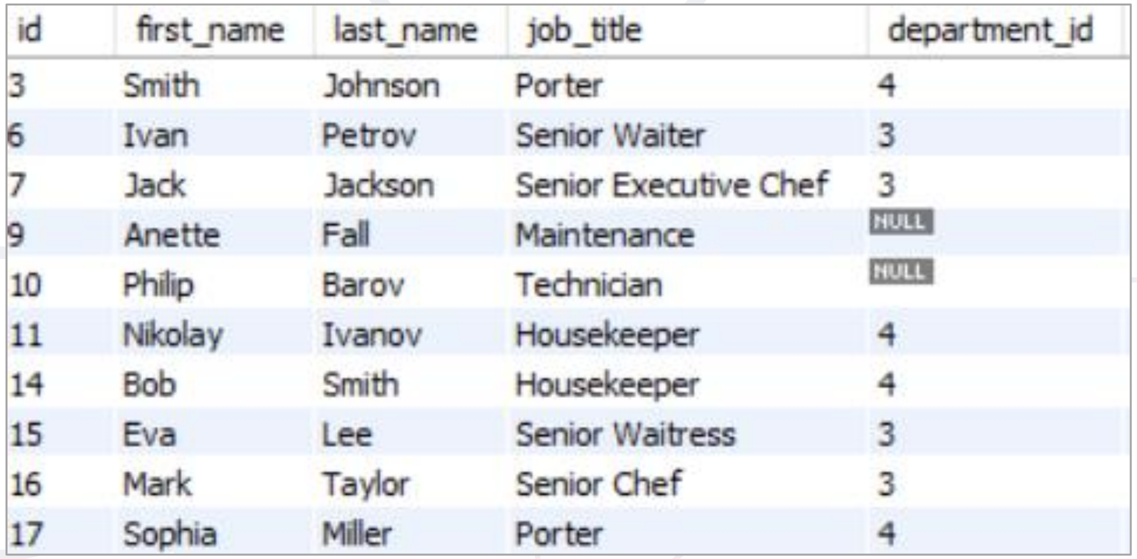
  - Note: Don't forget the **WHERE** clause!

- Delete all rows from a table (**TRUNCATE** works faster than **DELETE**)

```
TRUNCATE TABLE clients;
```

Problem: Delete from Table

- **Delete** all employees from the "**employees**" table who are in department **2 or 1**.
- **Order** the rest by id.



id	first_name	last_name	job_title	department_id
3	Smith	Johnson	Porter	4
6	Ivan	Petrov	Senior Waiter	3
7	Jack	Jackson	Senior Executive Chef	3
9	Anette	Fall	Maintenance	NULL
10	Philip	Barov	Technician	NULL
11	Nikolay	Ivanov	Housekeeper	4
14	Bob	Smith	Housekeeper	4
15	Eva	Lee	Senior Waitress	3
16	Mark	Taylor	Senior Chef	3
17	Sophia	Miller	Porter	4

Solution: Delete from Table

Delete Data

OR Condition

```
DELETE FROM employees
WHERE department_id = 1
OR department_id = 2;


SELECT * FROM employees;
```

SQL in Testing

Database Interaction for Effective Testing

Why SQL Is Important in Testing?

- **Central Role of Databases:** The backbone of virtually every system
- **Relational Database Management:** MySQL and Oracle are widely used for storing and organizing data
- **Standard Language for Data Processing:** SQL stands as the industry-standard computer language for relational database management and data processing
- **Accessing and Managing Data:** Crucial language for accessing and managing the data within the database
- **Data Manipulation and Retrieval:** Wide range of essential operations, including querying, inserting, updating, and modifying data

Must have knowledge for QAs

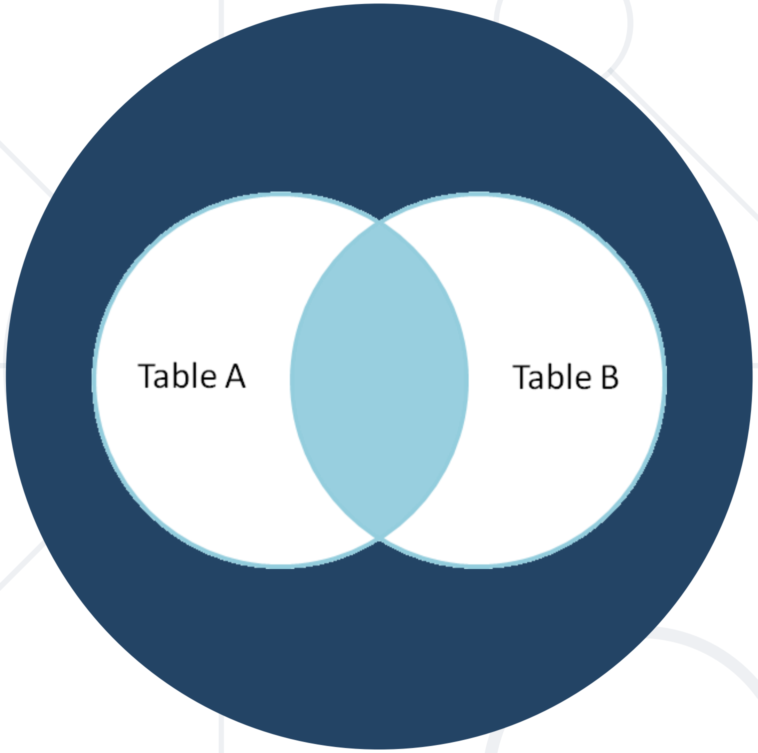
- Recognize **Various Types** of Databases
- Connect Using **Different SQL Clients**
- Comprehend Database **Relationships, Keys, and Indexes**
- Write **Simple and Complex** SQL Queries
- Perform **Data Validation** and Testing Techniques
- Test **Data Modifications** and Transactions
- **Explore** Database Schema
- **Interpret** Complex Queries

Really Complex Queries ;)

```
SELECT d.name AS department_name,
       COUNT(e.id) AS total_employees,
       AVG(e.salary) AS average_salary,
       MIN(e.salary) AS min_salary,
       MAX(e.salary) AS max_salary
FROM departments d
LEFT JOIN employees e ON d.id = e.department_id
GROUP BY d.name
HAVING total_employees >= 2
ORDER BY average_salary DESC;
```

Query Explanation:

- This SQL query performs the following tasks:
  - Retrieve **Department Information**
  - **Count Total Employees** per Department
  - **Calculate Average Salary** per Department
  - **Find Minimum** and **Maximum Salaries** per Department
  - **Group and Filter** the Results
  - The HAVING clause is used to **filter out departments** with fewer than two employees
  - **Sorts** the Results



SQL Practice

Master the Most Used SQL Statements!

Some of Most Used SQL Statements

- **CREATE** - Creates a new database, table, view, or other database objects
- **INSERT INTO** - Adds new rows (records) into a table to store data in the database
- **DROP** - Deletes an existing database object, such as a table, from the database
- **ALTER** - Modifies the structure of an existing database object, such as a table or column
- **UPDATE** – Updates specific records with new values
- **SELECT** - Specifies the database to be used for subsequent SQL statements

Some of Most Used SQL Statements

- **SELECT DISTINCT** - Extracts unique values from one or more fields in the result set, removing duplicates
- **WHERE** - Specifies which rows to retrieve based on specific conditions
- **IN** - Checks if a value matches any value in a specified list or subquery
- **BETWEEN** - Filters data within a specified range of values
- **LIKE** - Performs pattern matching to find specific values in the data
- **ORDER BY** - Sorts the result set in ascending or descending order based on specified columns
- **AND, OR** and **NOT** Operators

Some of Most Used SQL Statements

- **Aggregate Functions** - Perform calculations involving a range of values and returns a single value
  - MIN, MAX
  - AVG, SUM, COUNT
- HAVING - Added because the WHERE keyword cannot be used with aggregate functions
- INNER JOIN - Selects records that have matching values in both tables
- MySQL Operators – Arithmetic, Comparison, Compound, Logical
- Primary Key Constraint - uniquely identifies each record in a table
- Foreign Key Constraint - refers to the PRIMARY KEY in another table

Summary

- What are **Databases**?
- **SQL** and **MySQL** in short
- How to Retrieve Data – **SELECT**
- How to Write Data - **INSERT**
- How to Modify Data – **UPDATE, DELETE**
- Why SQL is **needed in Testing**?
- Further **Practice**

Questions?

SoftUni

Software University

SoftUni Creative

SoftUni Digital

SoftUni Foundation

SoftUni Kids

Finance Academy

SoftUni Diamond Partners

Software University

SUPER HOSTING .BG

Coca-Cola HBC Bulgaria

POKERSTARS
POKER | CASINO | SPORTS
a Flutter International brand

INDEAVR
Serving the high achievers

AMBITIONED

DRAFT KINGS

SOFTWARE GROUP

createX

Postbank
Решения за твоето утре

BOSCH

DXC TECHNOLOGY

SmartIT

Educational Partners

Software University

CODE IT UP

License

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg, about.softuni.bg
- Software University Foundation
  - softuni.foundation
- Software University @ Facebook
  - facebook.com/SoftwareUniversity
- Software University Forums
  - forum.softuni.bg