**National University of Computer & Emerging Sciences, Karachi** 

**Computer Science Department**

**Fall 2023, Lab Manual - 01**

|  |  |
| --- | --- |
| **Course Code: CL-2005** | **Course: Database Systems Lab** |
| **Instructor(s):** | **Syeda Mahnoor Javed** |

# Contents:

1. Database
2. SQL
3. Basic SQL Concepts

# Database

A database is a systematic collection of data. They support electronic storage and manipulation of data. Databases make data management easy.

**Example #1**

An online telephone directory uses a database to store data of people, phone numbers, and other contact details. Your electricity service provider uses a database to manage billing, client-related issues, handle fault data, etc.

**Example #2**

Facebook needs to store, manipulate, and present data related to members, their friends, member activities, messages, advertisements, and a lot more. We can provide a countless number of examples for the usage of databases.

# SQL

**SQL** is the standard language for dealing with Relational Databases. SQL can be used to insert, search, update, and delete database records. SQL can do lots of other operations, including optimizing and maintenance of databases. SQL stands for Structured Query language, pronounced as "S-Q-L" or sometimes as "See-Quel"... Relational databases like MySQL Database, Oracle, MS SQL Server, Sybase, etc. use ANSI SQL.

# Basic SQL Concepts

1. **Data Types**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| bigint | decimal | real | char | nvarchar |
| int | numeric | datetime | varchar | nvarchar(max) |
| smallint | money | smalldatetime | varchar(max) | ntext |
| tinyint | smallmoney | date | text | binary |
| bit | float | time | nchar | varbinary |
| varbinary(max) | image |  |  |  |

1. **Arithmetic operators**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Addition | Subtraction | Multiplication | Division | Modulus |
| **+** | **-** | **\*** | **/** | **%** |

1. **SQL Comparison Operators**

|  |  |
| --- | --- |
| **=** | Checks if the values of two operands are equal or not, if yes then condition becomes true. |
| **!=** | Checks if the values of two operands are equal or not, if values are not equal then condition becomes true. |
| **<>** | Checks if the values of two operands are equal or not, if values are not equal then condition becomes true. |
| **>** | Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true. |
| **<** | Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true. |
| **>=** | Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true. |
| **<=** | Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true. |
| **!<** | Checks if the value of left operand is not less than the value of right operand, if yes then condition becomes true. |
| **!>** | Checks if the value of left operand is not greater than the value of right operand, if yes then condition becomes true. |

1. **SQL Logical Operators**

|  |  |
| --- | --- |
| **ALL** | The ALL operator is used to compare a value to all values in another value set. |
| **AND** | The AND operator allows the existence of multiple conditions in an SQL statement's WHERE clause |
| **ANY** | The ANY operator is used to compare a value to any applicable value in the list as per the condition. |
| **BETWEEN** | The BETWEEN operator is used to search for values that are within a set of values, given the minimum value and the maximum value. |
| **EXISTS** | The EXISTS operator is used to search for the presence of a row in a specified table that meets a certain criterion. |
| **IN** | The IN operator is used to compare a value to a list of literal values that have been specified. |
| **LIKE** | The LIKE operator is used to compare a value to similar values using wildcard operators. |
| **NOT** | The NOT operator reverses the meaning of the logical operator with which it is used. Eg: NOT EXISTS, NOT BETWEEN, NOT IN, etc. **This is a negate operator.** |
| **OR** | The OR operator is used to combine multiple conditions in an SQL statement's WHERE clause. |
| **NULL** | The NULL operator is used to compare a value with a NULL value. |
| **UNIQUE** | The UNIQUE operator searches every row of a specified table for uniqueness (no duplicates). |

1. **Basic SQL Queries**

**Note: Connect the HR Database in SqlDeveloper**

Select \* from EMPLOYEES

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **EMPLOYEE\_ID** | **FIRST\_NAME** | **LAST\_NAME** | **EMAIL** | **PHONE\_NUMBER** | **HIRE\_DATE** | **JOB\_ID** |
| 100 | Steven | King | SKING | 515.123.4567 | 17-Jun-03 | AD\_PRES |
| 101 | Neena | Kochhar | NKOCHHAR | 515.123.4568 | 21-Sep-05 | AD\_VP |
| 102 | Lex | De Haan | LDEHAAN | 515.123.4569 | 13-Jan-01 | AD\_VP |
| -------- | -------- | -------- | -------- | -------- | -------- | -------- |
| 1023 | Lex3 | De Haanas | LDEsdaHAAN | 515.123.4569 | 13-Jan-01 | AD\_VPP |

Select EMPLOYEE\_ID, FIRST\_NAME, SALARY from EMPLOYEES

|  |  |  |
| --- | --- | --- |
| **EMPLOYEE\_ID** | **FIRST\_NAME** | **SALARY** |
| 100 | Steven | 24000 |
| 101 | Neena | 17000 |
| 102 | Lex | 17000 |
| ------------------------------------ | ------------------------------------ | --------------------------------------------------------- |
| 1023 | Lex3 | 12000 |

Select EMPLOYEE\_ID, FIRST\_NAME, SALARY from EMPLOYEES where salary>2300

|  |  |  |
| --- | --- | --- |
| **EMPLOYEE\_ID** | **FIRST\_NAME** | **SALARY** |
| 100 | Steven | 24000 |

Select EMPLOYEE\_ID, FIRST\_NAME, SALARY from EMPLOYEES where salary greater than or equal to 10000 and less than or equal to 12000

|  |  |  |
| --- | --- | --- |
| **EMPLOYEE\_ID** | **FIRST\_NAME** | **SALARY** |
| 114 | Den | 11000 |
| 147 | Alberto | 12000 |
| 148 | Gerald | 11000 |
| 149 | Eleni | 10500 |
| 114 | Den | 11000 |

**Let’s Practice More!!**

**Comparison operator:**

* SELECT \* FROM EMPLOYEES WHERE MANAGER\_ID = 101;
* SELECT \* FROM EMPLOYEES WHERE MANAGER\_ID < 110;
* SELECT \* FROM EMPLOYEES WHERE MANAGER\_ID > 200;
* SELECT \* FROM EMPLOYEES WHERE MANAGER\_ID >= 200;
* SELECT \* FROM EMPLOYEES WHERE MANAGER\_ID <= 150;
* ELECT \* FROM EMPLOYEES WHERE MANAGER\_ID <> 114;

**Logical Operators:**

* SELECT FIRST\_NAME,SALARY,JOB\_ID,DEPARTMENT\_ID FROM EMPLOYEES WHERE JOB\_ID = 'AD\_VP' AND DEPARTMENT\_ID = 90;
* SELECT FIRST\_NAME,SALARY,JOB\_ID,DEPARTMENT\_ID FROM EMPLOYEES WHERE JOB\_ID = 'AD\_VP' OR DEPARTMENT\_ID = 90;
* SELECT FIRST\_NAME,SALARY,JOB\_ID,DEPARTMENT\_ID FROM EMPLOYEES WHERE Not JOB\_ID = 'AD\_VP';

# LAB TASKS:

1. Display all columns from the `Jobs` table where the salary is less than $20,000.

2. Find the minimum and maximum salary for the job title 'Manager' from the `Jobs` table.

3. Find employees with salaries greater than or equal to $25,000 and display their employee ID, first name, and salary from the `Employees` table.

4. Find jobs with salaries greater than $12,000 and less than $25,000. Display the job titles and salaries.

5. Find employees whose last names are "Johnson", and return their employee ID, first name, and department ID.

6. Find employees working in departments 50 or 60, and return their employee ID, first name, and department ID.

7. Find the phone numbers of employees with a department ID of 80 and do not have a manager ID of 102.

8. Find employees with first names 'John', 'Alice', or 'Sandy', and display their employee ID and department ID.

9. Find cities from the `Locations` table where the country ID is 'IT'.

10. Find cities from the `Locations` table where the country ID is neither 'IN' nor 'CH'.

11. Find jobs with minimum salaries between 8,000 and 12,000, and include job title and maximum salary.

12. Find phone numbers of employees in department ID 90 but exclude those with job titles 'CLERK'.

13. Find employees hired after '01-Jul-2007', including their employee ID, first name, and hire date.

14. Find employees hired after '01-Jan-2008' in departments 100 or 120, and display their employee ID, name, and hire date.

15. Find employees hired before '01-Jan-2009' who are not in department 110, and display their employee ID, name, and department ID.

16. Find employees with a commission percentage of 0 and who do not work in department 90, and display their employee ID and commission percentage.

17. Find employees hired in the year 2009, including their employee ID, first name, and department ID.

18. Find jobs where the maximum salary is between 10,000 and 25,000, and include job title and minimum salary.

19. Find employees with IDs between 110 and 160 and department IDs between 80 and 100, and display their first name, last name, and job title.

20. Calculate the total salary including commission for each employee where the formula is `Salary + (Salary \* Commission\_Pct)`, and display the employee ID and total salary.