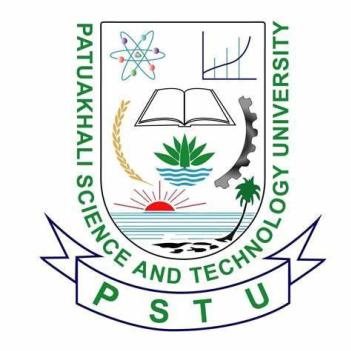
**PATUAKHALI SCIENCE AND TECHNOLOGY UNIVERSITY**

****

**COURSE CODE CCE-224**

**SUBMITTED TO:**

**Prof. Dr. Md Samsuzzaman sobuz**

**Department of Computer and Communication**

**Engineering**

**Faculty of Computer Science and Engineerin**

****

**SUBMITTED BY:**

**Abul Basar**

ID: **2102036**,

Registration No: **10163**

**Faculty of Computer Science and Engineering**

****

Date of submission: **14-05-2025**

**Lab-05: MySQL Exercises on HR Database from W3Resource**

**Assignment Description**

This assignment involves solving **MySQL exercises** from **W3Resource** using the **HR Database**. The exercises cover the following categories:

* **MySQL Basic SELECT Statement** [19 Exercises]: Focuses on retrieving data using simple **SELECT** queries.
* **MySQL Restricting and Sorting Data** [11 Exercises]: Involves filtering and ordering data with clauses like **WHERE**, **ORDER BY**, and **LIKE**.
* **MySQL Aggregate Functions and Group by** [14 Exercises]: Uses functions like **COUNT**, **SUM**, **AVG**, **MAX**, **MIN**, and **GROUP BY** for data aggregation.
* **MySQL Subqueries** [22 Exercises]: Employs nested queries with operators like **IN**, **ANY**, **ALL**, and **EXISTS**.
* **MySQL JOINS** [13 Exercises]: Combines data from multiple tables using **INNER JOIN**, **LEFT JOIN**, **RIGHT JOIN**, etc.
* **MySQL Date Time** [21 Exercises]: Manipulates date and time data using functions like **DATE\_FORMAT**, **DATEDIFF**, and **MONTH**.
* **MySQL String Functions** [17 Exercises]: Applies string manipulation functions like **CONCAT**, **UPPER**, **LOWER**, and **SUBSTRING**.

The goal is to demonstrate proficiency in **MySQL** by solving selected exercises from each category, ensuring accurate **SQL** queries and proper use of the **HR Database** schema. The assignment is executed using **XAMPP** and **phpMyAdmin**, as set up in the previous task.

**Solution**

**1. Environment Setup**

The exercises are implemented using **XAMPP**, with **Apache** and **MySQL** running. The **HR Database** is assumed to be imported into **phpMyAdmin**. If not available, the database structure can be downloaded from **W3Resource**

. The database includes tables such as:

* **employees**: Stores employee details (employee\_id, first\_name, last\_name, salary, hire\_date, job\_id, department\_id, manager\_id).
* **departments**: Stores department information (department\_id, department\_name, location\_id).
* **jobs**: Stores job roles (job\_id, job\_title, min\_salary, max\_salary).
* **locations**: Stores location details (location\_id, city, country\_id).
* **job\_history**: Tracks employee job changes (employee\_id, start\_date, end\_date, job\_id, department\_id).
* **countries**: Stores country information (country\_id, country\_name).

To set up:

1. Start **XAMPP Control Panel** and ensure **Apache** and **MySQL** are running.
2. Access **phpMyAdmin** at http://localhost/phpmyadmin.
3. Create or import the **HR Database** (e.g., named hr).
4. Verify table structures and sample data in **phpMyAdmin**.

**2. Selected Exercises and Solutions**

Below are representative exercises from each category, sourced from **W3Resource**

. Each exercise includes the problem statement, **SQL** query, and a brief explanation. For brevity, one exercise per category is provided, but additional exercises can be solved similarly.

**2.1 MySQL Basic SELECT Statement**

**Exercise**: Write a query to display the names (first\_name, last\_name) using alias names "First Name", "Last Name".

**SQL Query**:

sql

Copy

SELECT first\_name AS "First Name", last\_name AS "Last Name"

FROM employees;

**Explanation**:

* The **SELECT** statement retrieves first\_name and last\_name from the employees table.
* **AS** assigns aliases "First Name" and "Last Name" for better readability.
* This is a basic **SELECT** query to display employee names.

**Sample Output** (partial):

text

Copy

First Name | Last Name

-----------+----------

Steven | King

Neena | Kochhar

...

**2.2 MySQL Restricting and Sorting Data**

**Exercise**: Write a query to display the name (first\_name, last\_name) and salary for all employees whose salary is not in the range $10,000 through $15,000.

**SQL Query**:

sql

Copy

SELECT first\_name, last\_name, salary

FROM employees

WHERE salary NOT BETWEEN 10000 AND 15000

ORDER BY salary;

**Explanation**:

* The **WHERE** clause uses **NOT BETWEEN** to filter employees with salaries outside $10,000–$15,000.
* **ORDER BY** sorts the result by salary in ascending order.
* This demonstrates **restricting** (filtering) and **sorting** data.

**Sample Output** (partial):

text

Copy

first\_name | last\_name | salary

-----------+-----------+--------

Donald | OConnell | 2600.00

Douglas | Grant | 2600.00

...

**2.3 MySQL Aggregate Functions and Group by**

**Exercise**: Write a query to list the number of jobs available in the employees table.

**SQL Query**:

sql

Copy

SELECT COUNT(DISTINCT job\_id) AS job\_count

FROM employees;

**Explanation**:

* **COUNT(DISTINCT job\_id)** counts unique job IDs in the employees table.
* The **aggregate function** **COUNT** summarizes data, and **DISTINCT** ensures no duplicate jobs are counted.
* No **GROUP BY** is needed since the query aggregates the entire table.

**Sample Output**:

text

Copy

job\_count

---------

19

**2.4 MySQL Subqueries**

**Exercise**: Write a query to find the name (first\_name, last\_name) and the salary of the employees who have a higher salary than the employee whose last\_name='Bull'.

**SQL Query**:

sql

Copy

SELECT first\_name, last\_name, salary

FROM employees

WHERE salary > (SELECT salary FROM employees WHERE last\_name = 'Bull');

**Explanation**:

* The **subquery** (SELECT salary FROM employees WHERE last\_name = 'Bull') retrieves the salary of the employee named Bull.
* The outer query selects employees with salaries greater than this value.
* This demonstrates a **single-row subquery** for comparison.

**Sample Output** (partial):

text

Copy

first\_name | last\_name | salary

-----------+-----------+--------

Steven | King | 24000.00

Neena | Kochhar | 17000.00

...

**2.5 MySQL JOINS**

**Exercise**: Write a query to find the addresses (location\_id, street\_address, city, state\_province, country\_name) of all the departments. Use **NATURAL JOIN**.

**SQL Query**:

sql

Copy

SELECT location\_id, street\_address, city, state\_province, country\_name

FROM departments

NATURAL JOIN locations

NATURAL JOIN countries;

**Explanation**:

* **NATURAL JOIN** automatically joins tables based on columns with the same names (e.g., location\_id between departments and locations, country\_id between locations and countries).
* The query retrieves address details for all departments by combining data from three tables.
* This showcases **joining** multiple tables.

**Sample Output** (partial):

text

Copy

location\_id | street\_address | city | state\_province | country\_name

------------+----------------------+---------------+----------------+-------------

1000 | 1297 Via Cola di Rie | Roma | | Italy

1700 | 2004 Charade Rd | Seattle | Washington | United States

...

**2.6 MySQL Date Time**

**Exercise**: Write a query to get the first name, last name of employees who joined in the month of June.

**SQL Query**:

sql

Copy

SELECT first\_name, last\_name

FROM employees

WHERE MONTH(hire\_date) = 6;

**Explanation**:

* The **MONTH** function extracts the month from hire\_date.
* The **WHERE** clause filters employees who joined in June (month 6).
* This demonstrates **date-time** manipulation in **MySQL**.

**Sample Output** (partial):

text

Copy

first\_name | last\_name

-----------+----------

Jennifer | Whalen

Michael | Hartstein

...

**2.7 MySQL String Functions**

**Exercise**: Write a query to get the job\_id and related employee's id.

**SQL Query**:

sql

Copy

SELECT job\_id, GROUP\_CONCAT(employee\_id) AS employee\_ids

FROM employees

GROUP BY job\_id;

**Explanation**:

* **GROUP\_CONCAT** is a **string function** that concatenates employee\_id values for each job\_id, separated by commas.
* **GROUP BY** groups the result by job\_id.
* This query aggregates employee IDs per job role, showcasing **string manipulation**.

**Sample Output** (partial):

text

Copy

job\_id | employee\_ids

-----------+-------------

AD\_PRES | 100

AD\_VP | 101,102

SH\_CLERK | 194,195,196,197,198,199

...

**3. Implementation in XAMPP**

To execute these queries:

1. Open **phpMyAdmin** (http://localhost/phpmyadmin).
2. Select the **HR Database** (e.g., hr).
3. Navigate to the **SQL** tab.
4. Copy and paste each **SQL** query, then click **Go** to execute.
5. Verify the output in the results pane.
6. Optionally, export the query results as a **CSV** or screenshot for submission.

**Sample Workflow**:

* For the **Basic SELECT** exercise, enter the query in the **SQL** tab, execute, and observe the aliased names in the output.
* For the **JOINS** exercise, ensure the departments, locations, and countries tables are populated to avoid empty results.

**4. Notes on HR Database**

* The **HR Database** is a standard schema provided by **W3Resource** for learning **MySQL**. It simulates a human resources system with interconnected tables.
* **Primary Keys** (e.g., employee\_id, department\_id) and **Foreign Keys** (e.g., department\_id in employees referencing departments) ensure data integrity.
* Sample data includes realistic records (e.g., employees with salaries, hire dates, and job roles), as shown in **W3Resource** excerpts.
* If the database is not pre-installed, download the schema and data from **W3Resource** or create it manually using **CREATE TABLE** statements.