|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **National University of Computer and Emerging Sciences, Lahore Campus** | | | | |
| C:\Users\saif\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.Word\final design.jpg | **Course:** | **Data Base (Lab)** | **Course Code:** | **CS-2005** |
| **Program:** | **BS (Computer Science)** | **Semester:** | **Spring 2023** |
| **Duration:** | **2.5 Hrs** | **Total Marks:** | **55** |
| **Quiz Date:** | **12-June-23** | **Weight** | **\_40\_** |
| **Section:** | **BCS-4E, BCS-4F, BCS-4G, BCS-4H, BCS- 4J, BSE- 6A, BSE- 6B [SLOT 1]** | **Page(s):** | **2** |
| **Name** |  | **Roll No.** |  |
|  |  | | | |

**Instruction/Notes:**

1. Use the schema and final exam from the folder: \\cactus1\Xeon\Spring 2023\Samman Ashraf\Database Lab Final.
2. Use of internet is not allowed.
3. **Submission Format:**
   1. Make a folder with your Registration No and paste TWO files (Word & SQL) in it.
   2. **Word File:** Queries should be pasted in this given file along with their screenshots of running queries (these are must otherwise marks will be deducted). Submit this word file named as (L13-4632.docx ).
   3. **SQL File:** It includes all your .sql queries of the given questions. (e.g; yourName.sql)
4. Submit your final paper in the given folder. \\cactus1\Xeon\Spring 2023\Samman Ashraf\Database Final Submissions\Your Section (e.g; ABC-4K).
5. **Attempt all questions carefully. Your marking is Binary. Please Note SS is Screenshots.**

**Use CompanyDB for the paper**

**Question no 1:** **[5 (1.5 SS) + 5 (1.5 SS)= 10 Marks]**

1. Write a SQL query to retrieve the total hours worked by employees in each department, along with the department name and the employee with the highest hours in each department.

|  |
| --- |
| **SELECT**  **D.name AS department\_name,**  **SUM(TE.hours) AS total\_hours,**  **(**  **SELECT TOP 1**  **E.name**  **FROM**  **employees E**  **JOIN**  **time\_entries TE ON E.id = TE.employee\_id**  **WHERE**  **E.department\_id = D.id**  **ORDER BY**  **TE.hours DESC**  **) AS employee\_with\_highest\_hours**  **FROM**  **departments D**  **JOIN**  **employees E ON D.id = E.department\_id**  **JOIN**  **time\_entries TE ON E.id = TE.employee\_id**  **GROUP BY**  **D.name;** |

1. Write a SQL query which retrieves the names of employees and their corresponding department names for employees who have worked on all tasks in their department. **(Nested Query)**

|  |
| --- |
| **SELECT e.name AS employee\_name, d.name AS department\_name**  **FROM employees e**  **JOIN departments d ON e.department\_id = d.id**  **WHERE e.department\_id IN (**  **SELECT department\_id**  **FROM tasks**  **GROUP BY department\_id**  **HAVING COUNT(DISTINCT id) = (**  **SELECT COUNT(DISTINCT task\_id)**  **FROM time\_entries**  **WHERE time\_entries.department\_id = tasks.department\_id**  **AND time\_entries.employee\_id = e.id**  **)**  **);** |

**Question no 2: [7.5 (1.5 SS) + 7.5 (1.5 SS)= 15 Marks]**

1. Create a view named "EmployeeSummary" that displays the top 2 employee name, department name, and the total number of hours worked by each employee. Include only those employees who have worked at least 20 hours in total. Order the results in descending order of the total number of hours worked. Create a view named "EmployeeSummary" that displays the top 2 employee name, department name, and the total number of hours worked by each employee. Include only those employees who have worked at least 20 hours in total. Order the results in descending order of the total number of hours worked.

|  |
| --- |
| **CREATE VIEW EmployeeSummary AS SELECT TOP 2 e.name AS employee\_name, d.name AS department\_name, SUM(te.hours) AS total\_hours\_worked FROM employees e JOIN departments d ON e.department\_id = d.id JOIN time\_entries te ON e.id = te.employee\_id GROUP BY e.id, e.name, d.name HAVING SUM(te.hours) >= 20 ORDER BY total\_hours\_worked DESC;**  **select \* from EmployeeSummary;** |

1. Create a view “TimeEntrySummary” to get a summary of time entries by employee and task.

|  |
| --- |
| **CREATE VIEW TimeEntrySummary AS**  **SELECT**  **E.name AS employee\_name,**  **T.name AS task\_name,**  **SUM(TE.hours) AS total\_hours**  **FROM**  **time\_entries TE**  **JOIN**  **employees E ON TE.employee\_id = E.id**  **JOIN**  **tasks T ON TE.task\_id = T.id**  **GROUP BY**  **E.name, T.name;** |

**Question no 3: [7.5 (1.5 SS) + 7.5 (1.5 SS)= 15 Marks]**

1. Create a stored procedure named GetDepartmentTasks that accepts a department ID as a parameter and returns all the tasks associated with that department, along with the total number of employees assigned to each task

|  |
| --- |
| **CREATE PROCEDURE GetDepartmentTasks @departmentId INT**  **AS BEGIN**  **SELECT t.name AS task\_name, COUNT(e.id) AS total\_employees**  **FROM tasks t LEFT JOIN employees e ON t.department\_id = e.department\_id**  **WHERE t.department\_id = @departmentId GROUP BY t.name;**  **END**  **EXEC GetDepartmentTasks @departmentId = 1;** |

1. Create a stored procedure that calculates the average hours worked per task for a specific department:

|  |
| --- |
| **CREATE PROCEDURE CalculateAverageHoursPerTask**  **@department\_id INT**  **AS**  **BEGIN**  **SELECT**  **T.name AS task\_name,**  **AVG(TE.hours) AS average\_hours**  **FROM**  **tasks T**  **JOIN**  **time\_entries TE ON T.id = TE.task\_id**  **JOIN**  **employees E ON TE.employee\_id = E.id**  **WHERE**  **E.department\_id = @department\_id**  **GROUP BY**  **T.name;**  **END;**  **EXEC CalculateAverageHoursPerTask @department\_id = 1;** |

**Question no 4: [7.5 (1.5 SS) + 7.5 (1.5 SS)= 15 Marks]**

1. Create a trigger that automatically updates the "updated\_at" column of the "employees" table whenever a row is updated.

|  |
| --- |
| **CREATE TRIGGER update\_employee\_updated\_at**  **BEFORE UPDATE ON employees**  **FOR EACH ROW**  **BEGIN**  **SET NEW.updated\_at = CURRENT\_TIMESTAMP;**  **END;** |

1. Create a trigger that prevents the deletion of a department if there are employees associated with it.

|  |
| --- |
| **CREATE TRIGGER prevent\_department\_deletion**  **BEFORE DELETE ON departments**  **FOR EACH ROW**  **BEGIN**  **DECLARE employee\_count INT;**  **SELECT COUNT(\*) INTO employee\_count FROM employees WHERE department\_id = OLD.id;**  **IF employee\_count > 0 THEN**  **SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Cannot delete department with associated employees';**  **END IF;**  **END;** |

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*