SQL_queries_problems

May 24, 2024

0.0.1 Basic SQL Queries

1. **Select All Columns From a Table**

- Select all columns from the `employees` table.

2. **Select Specific Columns**

- Select the `name` and `salary` columns from the `employees` table.

3. **Distinct Values**

- Find distinct job titles in the `employees` table.

4. **Where Clause**

- Select all employees who have a salary greater than 50,000.

5. **AND, OR Clauses**

- Select all employees who have a salary greater than 50,000 and work in the ’HR’ department.

6. **Order By**

- Select all employees ordered by their `hire_date` in descending order.

7. **Limit Clause**

- Select the top 5 highest paid employees.

8. **Between Clause**

- Select all employees who were hired between ’2020-01-01’ and ’2021-01-01’.

9. **IN Clause**

- Select all employees who work in the ’HR’, ’Finance’, or ’IT’ departments.

10. **LIKE Clause**
    - Select all employees whose names start with ’A’.

0.0.2 Aggregate Functions

11. **COUNT Function**
    - Count the number of employees in the `employees` table.
12. **SUM Function**

- Calculate the total salary of all employees.

13. **AVG Function**
    - Calculate the average salary of all employees.
14. **MIN Function**
    - Find the minimum salary in the `employees` table.
15. **MAX Function**
    - Find the maximum salary in the `employees` table.
16. **GROUP BY**
    - Group employees by department and count the number of employees in each department.
17. **HAVING Clause**
    - Find departments with more than 10 employees.
18. **GROUP BY with Aggregate Functions**
    - Find the average salary for each department.

0.0.3 Joins

19. **Inner Join**
    - Select all employees and their respective department names.
20. **Left Join**
    - Select all employees and their respective department names, including those without a department.
21. **Right Join**
    - Select all departments and their respective employees, including those without employees.
22. **Full Outer Join**
    - Select all employees and departments, including those without matches.
23. **Self Join**
    - Find pairs of employees who have the same manager.
24. **Cross Join**
    - Generate all possible combinations of employees and departments.
25. **Join on Multiple Conditions**
    - Select all employees and their projects, including the project start date and end date.

0.0.4 Subqueries

26. **Subquery in SELECT**
    - Select the name of each employee and their department's total salary.
27. **Subquery in FROM**
    - Select the average salary of departments with an average salary greater than 60,000.
28. **Subquery in WHERE**
    - Select all employees who have a salary greater than the average salary of their department.
29. **Correlated Subquery**
    - Select all employees who have the highest salary in their department.
30. **Subquery with EXISTS**

- Select all employees who have been assigned at least one project.

0.0.5 String Functions

31. **Concatenate**
    - Concatenate the first name and last name of employees.
32. **Substring**
    - Select the first three characters of the `name` column for all employees.
33. **Length**
    - Find the length of each employee's name.
34. **UPPER and LOWER**
    - Convert all employee names to uppercase.
35. **TRIM**
    - Trim leading and trailing spaces from employee names.
36. **REPLACE**
    - Replace all occurrences of 'Manager' with 'Team Lead' in the job title column.

0.0.6 Date Functions

37. **Current Date**
    - Select the current date.
38. **Date Difference**
    - Calculate the number of days each employee has been with the company.
39. **Extract Year**
    - Extract the year from the `hire_date` of each employee.
40. **Extract Month**
    - Extract the month from the `hire_date` of each employee.
41. **Date Add**
    - Add 1 year to the `hire_date` of each employee.
42. **Date Subtract**
    - Subtract 1 year from the `hire_date` of each employee.

0.0.7 Advanced SQL Queries

43. **CASE Statement**
    - Categorize employees into salary ranges (e.g., 'Low', 'Medium', 'High').
44. **Pivot**
    - Pivot the data to show departments as columns and count of employees in each department.
45. **Unpivot**
    - Unpivot the data to convert columns into rows.
46. **Recursive CTE**
    - Generate a list of all employees in a hierarchy with their managers.
47. **Window Functions**
    - Rank employees based on their salary within each department.

48. **Common Table Expressions (CTE)**
    - Use CTE to simplify complex queries for calculating the average salary in each department.
49. **Rank**
    - Rank employees by salary within their department.
50. **Row Number**
    - Assign a unique row number to each employee within their department ordered by hire date.

0.0.8 Data Modification

51. **Insert**
    - Insert a new employee into the `employees` table.
52. **Update**
    - Update the salary of all employees in the 'HR' department by 10%.
53. **Delete**
    - Delete all employees who have not been assigned a department.
54. **Upsert (Insert or Update)**
    - Insert a new employee or update the existing employee if they already exist.

0.0.9 Set Operations

55. **UNION**
    - Combine the results of two queries to list all employees and managers.
56. **UNION ALL**
    - Combine the results of two queries including duplicates.
57. **INTERSECT**
    - Find common employees who are both in the `employees` and `managers` tables.
58. **EXCEPT**
    - Find employees who are in the `employees` table but not in the `managers` table.

0.0.10 Indexes and Performance

59. **Create Index**
    - Create an index on the `name` column of the `employees` table.
60. **Drop Index**
    - Drop the index on the `name` column of the `employees` table.
61. **Query Optimization**
    - Optimize a query to select employees with a salary greater than 60,000.

0.0.11 Data Types and Constraints

62. **Check Constraint**
    - Add a check constraint to ensure that employee salaries are always greater than 30,000.

63. **Default Value**
    - Add a default value for the `hire_date` column to be the current date.
64. **Not Null Constraint**
    - Ensure that the `name` column in the `employees` table cannot be null.
65. **Unique Constraint**
    - Add a unique constraint on the `email` column in the `employees` table.

0.0.12 Complex Queries and Analysis

66. **Top-N Analysis**
    - Select the top 3 highest paid employees in each department.
67. **Duplicate Records**
    - Find duplicate records in the `employees` table based on the `email` column.
68. **Nth Highest Salary**
    - Find the 5th highest salary in the `employees` table.
69. **Gaps and Islands**
    - Identify continuous periods of employment in the `employees` table.
70. **Moving Average**
    - Calculate the moving average of salaries over a 3-month period.

0.0.13 Transaction and Locking

71. **Begin Transaction**
    - Start a transaction to update employee salaries and commit the changes.
72. **Rollback Transaction**
    - Start a transaction to update employee salaries and rollback the changes if an error occurs.
73. **Deadlock Analysis**
    - Analyze and resolve a deadlock situation in the database.
74. **Isolation Levels**
    - Set the transaction isolation level to `READ COMMITTED`.

0.0.14 JSON and XML

75. **JSON Functions**
    - Extract data from a JSON column in the `employees` table.
76. **XML Functions**
    - Extract data from an XML column in the `employees` table.

0.0.15 Miscellaneous

77. **Dynamic SQL**
    - Write a dynamic SQL query to select columns based on user input.
78. **Stored Procedures**

    - Create a stored procedure to insert a new employee.

79. **Triggers**
    - Create a trigger to log changes to the `employees` table.
80. **Views**
    - Create a view to simplify access to the employee details.
81. **User Defined Functions**
    - Create a user-defined function to calculate the annual salary of an employee.
82. **Partitioning**
    - Partition the `employees` table by department.
83. **Foreign Key Constraint**
    - Add a foreign key constraint between `employees` and `departments` tables.
84. **Cascade Delete**
    - Implement a cascade delete between `employees` and `departments`.
85. **Self-Referencing Foreign Key**
    - Add a self-referencing foreign key to track employee-manager relationships.

0.0.16 Analytical Queries

86. **Percentile**
    - Calculate the 90th percentile salary in the `employees` table.
87. **Cumulative Sum**
    - Calculate the cumulative sum of salaries ordered by `hire_date`.
88. **Lag Function**
    - Use the `LAG` function to compare each employee's salary with the previous one.
89. **Lead Function**
    - Use the `LEAD` function to compare each employee's salary with the next one.

0.0.17 Data Cleanup and Transformation

90. **Remove Duplicates**

- Remove duplicate records from the `employees` table.

91. **Normalize Data**
    - Normalize the `employees` table to the 3rd normal form.
92. **Denormalize Data**
    - Denormalize the `employees` table for faster querying.
93. **Data Masking**
    - Mask sensitive information in the `employees` table.

0.0.18 Security and Permissions

94. **Grant Permissions**
    - Grant read permissions on the `employees` table to a specific user.
95. **Revoke Permissions**
    - Revoke all permissions on the `employees` table from a specific user.

96. **Role-Based Access Control**
    - Implement role-based access control for the `employees` table.

0.0.19 Data Import and Export

97. **Import Data**
    - Import data from a CSV file into the `employees` table.
98. **Export Data**
    - Export data from the `employees` table to a CSV file.

0.0.20 Advanced Analytics

99. **Time Series Analysis**
    - Perform a time series analysis on employee hire dates.
100. **Geospatial Analysis**
    - Perform a geospatial analysis to find the distance between employee locations.

[ ]: