Vanessa Chen
HHA 507 - Oracle SQL Training
Dr. Hants Williams

Database Programming with SQL 9-1: Using GROUP BY and HAVING Clauses Practice Activities

- *Vocabulary:* Identify the vocabulary word for each definition below
  - Used to specify which groups are to be displayed; restricts groups that do not meet group criteria
    - Answer: HAVING
  - Divides the rows in a table into groups
    - Answer: GROUP BY
- *Try It / Solve It*
  - 1. In the SQL query shown below, which of the following is true about this query?
    - SELECT last_name, MAX(salary)
      FROM employees
      WHERE last_name LIKE 'K%'
      GROUP BY manager_id, last_name
      HAVING MAX(salary) >16000
      ORDER BY last_name DESC ;
      - **a. Kimberly Grant would not appear in the results set**
      - b. The GROUP BY clause has an error because the manager_id is not listed in the SELECT clause
      - **c. Only salaries greater than 16001 will be in the result set**
      - d. Names beginning with Ki will appear after names beginning with Ko
      - e. Last names such as King and Kochhar will be returned even if they don't have salaries > 16000
  - 2. Each of the following SQL queries has an error. Find the error and correct it. Use Oracle Application Express to verify that your corrections produce the desired results
    - a. SELECT manager_id
      FROM employees
      WHERE AVG(salary) < 16000
      GROUP BY manager_id;
      - SELECT manager_id
        FROM employees
        GROUP BY manager_id
        HAVING AVG(salary) < 16000;
    - b. SELECT cd_number, COUNT(title)

FROM d_cds
WHERE cd_number < 93;

- SELECT cd_number, COUNT(title)
FROM d_cds
WHERE cd_number < 93
GROUP BY cd_number;

- c. SELECT ID, MAX(ID), artist AS Artist
FROM d_songs
WHERE duration IN('3 min', '6 min', '10 min') HAVING ID < 50
GROUP BY ID;
  - SELECT ID, MAX(ID), artist AS Artist
  FROM d_songs
  WHERE duration IN('3 min', '6 min', '10 min') AND ID < 50
  GROUP BY ID, artist;
- d. SELECT loc_type, rental_fee AS Fee
FROM d_venues
WHERE id <100
GROUP BY "Fee"
ORDER BY 2;
  - SELECT loc_type, rental_fee AS Fee
  FROM d_venues
  WHERE id <100
  GROUP BY loc_type, rental_fee
  ORDER BY 2;

- 3. Rewrite the following query to accomplish the same result:
  - SELECT DISTINCT MAX(song_id)
  FROM d_track_listings
  WHERE track IN (1, 2, 3);
    - SELECT DISTINCT MAX(song_id)
    FROM d_track_listings
    WHERE track IN (1, 2, 3)
    GROUP BY track;
- 4. Indicate True or False
  - a. If you include a group function and any other individual columns in a SELECT clause, then each individual column must also appear in the GROUP BY clause
    - Answer: True
  - b. You can use a column alias in the GROUP BY clause
    - Answer: False
  - c. The GROUP BY clause always includes a group function

        - Answer: False
    - 5. Write a query that will return both the maximum and minimum average salary grouped by department from the employees table

        -
          ```
          SELECT MAX(dept_avg_salary) AS max_avg_salary,
                 MIN(dept_avg_salary) AS min_avg_salary
          FROM (
             SELECT department_id, AVG(salary) AS dept_avg_salary
             FROM employees
             GROUP BY department_id
          ) avg_salaries;
          ```
    - 6. Write a query that will return the average of the maximum salaries in each department for the employees table

        -
          ```
          SELECT AVG(max_salary) AS avg_max_salary
          FROM (
             SELECT MAX(salary) AS max_salary
             FROM employees
             GROUP BY department_id
          ) max_salaries;
          ```

Database Programming with SQL 9-2: Using ROLLUP and CUBE Operations and GROUPING SETS Practice Activities

- *Vocabulary:* Identify the vocabulary word for each definition below
    - Used to create subtotals that roll up from the most detailed level to a grand total, following a grouping list specified in the clause
        - Answer: ROLLUP
    - An extension to the GROUP BY clause like ROLLUP that produces cross-tabulation reports
        - Answer: CUBE
    - Used to specify multiple groupings of data
        - Answer: GROUPING SETS
- *Try It / Solve It*
    - 1. Within the Employees table, each manager_id is the manager of one or more employees who each have a job_id and earn a salary. For each manager, what is the total salary earned by all of the employees within each job_id? Write a query to display the Manager_id, job_id, and total salary. Include in the result the subtotal salary for each manager and a grand total of all salaries

        -
          ```
          SELECT manager_id, job_id, SUM(salary) AS total_salary
          FROM Employees
          GROUP BY ROLLUP (manager_id, job_id)
          ORDER BY manager_id, job_id;
          ```

- 2. Amend the previous query to also include a subtotal salary for each job_id regardless of the manager_id
  - SELECT manager_id, job_id, SUM(salary) AS total_salary
    FROM Employees
    GROUP BY
    GROUPING SETS ((manager_id, job_id), (manager_id), (job_id))
    ORDER BY manager_id, job_id;
- 3. Using GROUPING SETS, write a query to show the following groupings:
  - department_id, manager_id, job_id
  - manager_id, job_id
  - department_id, manager_ida
  - SELECT department_id, manager_id, job_id, SUM(salary) AS total_salary
    FROM Employees
    GROUP BY
    GROUPING SETS ((department_id, manager_id, job_id), (manager_id, job_id), (department_id, manager_id))
    ORDER BY department_id, manager_id, job_id;

Database Programming with SQL 9-3: Set Operators Practice Activities

- *Vocabulary:* Identify the vocabulary word for each definition below
  - operator that returns all rows from both tables and eliminates duplicates
    - Answer: UNION
  - columns that were made up to match queries in another table that are not in both tables
    - Answer: TO_CHAR(NULL)
  - operator that returns all rows from both tables, including duplicates
    - Answer: UNION ALL
  - used to combine results into one single result from multiple SELECT statements
    - Answer: SET operators
  - operator that returns rows that are unique to each table
    - Answer: MINUS
  - operator that returns rows common to both tables
    - Answer: INTERSECT
- *Try It / Solve It*
  - 1. Name the different Set operators?
    - UNION, UNION ALL, INTERSECT, MINUS
  - 2. Write one query to return the employee_id, job_id, hire_date, and department_id of all employees and a second query listing employee_id, job_id,

start_date, and department_id from the job_history table and combine the results as one single output. Make sure you suppress duplicates in the output

  - SELECT employee_id, job_id, hire_date, department_id
    FROM employees
    UNION
    SELECT employee_id, job_id, start_date AS hire_date, department_id
    FROM job_history;

- 3. Amend the previous statement to not suppress duplicates and examine the output. How many extra rows did you get returned and which were they? Sort the output by employee_id to make it easier to spot
  - SELECT employee_id, job_id, hire_date, department_id
    FROM employees
    UNION ALL
    SELECT employee_id, job_id, start_date AS hire_date, department_id
    FROM job_history
    ORDER BY employee_id;
- 4. List all employees who have not changed jobs even once. (Such employees are not found in the job_history table)
  - SELECT employee_id, job_id, hire_date, department_id
    FROM employees
    WHERE employee_id NOT IN (SELECT employee_id FROM job_history);
- 5. List the employees that HAVE changed their jobs at least once
  - SELECT DISTINCT employee_id, job_id, start_date AS hire_date, department_id
    FROM job_history;
- 6. Using the UNION operator, write a query that displays the employee_id, job_id, and salary of ALL present and past employees. If a salary is not found, then just display a 0 (zero) in its place
  - SELECT employee_id, job_id, salary
    FROM employees
    UNION
    SELECT employee_id, job_id, 0 AS salary
    FROM job_history;