9-1: Using GROUP BY and HAVING Clauses

1. In the SQL query shown below, which of the following is true about this query?
    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.

None of the above

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) < 93;
        i. SELECT manager_id FROM employees GROUP BY manager_id HAVINGAVG(salary) < 93;
    b. SELECT cd_number, COUNT(title) FROM d_cds WHERE cd_number < 93;
        i. SELECT cd_number, COUNT(title) FROM d_cds WHERE cd_number < 93GROUP 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;
        i. SELECT MAX(ID) AS max_id, artist AS Artist FROM d_songs WHERE duration IN ('3 min', '6 min', '10 min') GROUP BY artist HAVING MAX(ID) < 50;
    d. SELECT loc_type, rental_fee AS Fee FROM d_venues WHERE id
        i. SELECT loc_type, rental_fee AS Fee FROM d_venues WHERE id IS NOTNULL;

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);
    a. SELECT MAX(song_id)
    b. FROM f_track_listings
    c. WHERE track IN (1, 2, 3);
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.
        i. TRUE
    b. You can use a column alias in the GROUP BY clause
        i. FALSE
    c. The GROUP BY clause always includes a group function
        i. FALSE
5. Write a query that will return both the maximum and minimum average salary grouped by department from the employees table

    a. SELECT MAX(AVG(salary)) AS max_avg_salary,
        i. MIN((AVG(salary)) AS max_avg_salary
    b. FROM employees
    c. GROUP BY department_id;

6. Write a query that will return the average of the maximum salaries in each department for the employees table.
    a. SELECT AVG(MAX(salary)) AS max_avg_salary
    b. FROM employees
    c. GROUP BY department_id;

9-2: Using ROLLUP and CUBE Operations and GROUPING SETS

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. (You can use the ROLLUP operation to get the total salary for each manager_id and job_id, along with the subtotals for each manager_id and a grand total)
    a. SELECT manager_id, job_id, SUM(salary) AS total_salary
    b. FROM employees
    c. GROUP BY ROLLUP (manager_id, job_id);
2. Amend the previous query to also include a subtotal salary for each job_id regardless of the manager_id.
    a. SELECT manager_id, job_id, SUM (salary) AS total_salary
    b. FROM employees
    c. Group BY CUBE (manager_id, job_id);
3. Using GROUPING SETS, write a query to show the following groupings:
    a. department_id, manager_id, job_id
    b. manager_id, job_id
    c. department_id, manager_id
        i. SELECT depart_id, manager_id, job_id, SUM(salary) AS total_salary
        ii. FROM employees
        iii. GROUP BY GROUPING SETS (
            1. (department_id, manager_id, job_id
            2. (manager_id, job_id),
            3. (department_id, manager_id));

9-3: Set Operators

1. Name the different Set operators?
    a. UNION:
        i. Combines results of two queries, removing duplicates.
    b. UNION ALL:
        i. Combines results of two queries, including duplicates.

    c. INTERSECT:
        i. Returns only the rows that are common between two queries.
    d. MINUS:
        i. Returns the rows from the first query that are not present in the second query

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.
    a. SELECT employee_id, job_id, hire_date AS start_date, department_id
    b. FROM employees
    c. UNION
    d. SELECT employee_id, job_id, start_date, department_id
    e. 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.
    a. SELECT employee_id, job_id, hire_date AS start_date, department_id
    b. FROM employees
    c. UNION ALL
    d. SELECT employee_id, job_id, start_date, department_id
    e. FROM job_history;
    f. 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)
    a. SELECT e.emplyee_id, e.job_id, e.hire_date, e.department_id
    b. FROM employees e
    c. WHERE NOT EXISTS (
        i. SELECT 1
        ii. FROM job_history jh
        iii. WHERE e.employee_id = jh.employee_id);
5. List the employees that HAVE changed their jobs at least once.
    a. SELECT e.emplyee_id, e.job_id, e.hire_date, e.department_id
    b. FROM employees e
    c. WHERE EXISTS (
        i. SELECT 1
        ii. FROM job_history jh
        iii. WHERE e.employee_id = jh.employee_id);
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
    a. SELECT e.emplyee_id, e.job_id, e.hire_date, e.department_id
    b. FROM employees e
    c. UNION

i. SELECT employee_id, job_id 0 AS salary
ii. FROM job_history jh