Database Programming with SQL

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.

```
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 ;
```

None of the statements are True

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;

```
SELECT manager_id FROM employees GROUP BY manager_id
HAVING AVG(salary) < 93;
```

    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 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

SELECT loc_type, rental_fee AS Fee FROM d_venues WHERE id IS NOT NULL;

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);

```
1  SELECT MAX(song_id)
2  FROM d_track_listings
3  WHERE track IN (1, 2, 3);
```

Results | Explain | Describe | Saved SQL | History

MAX(SONG_ID)
49

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. **(True)**
   b. You can use a column alias in the GROUP BY clause. **(False)**
   c. The GROUP BY clause always includes a group function. **(False)**
5. Write a query that will return both the maximum and minimum average salary grouped by department from the employees table.

```
1  SELECT MAX(AVG(salary)) AS max_avg_salary,
2         MIN(AVG(salary)) AS min_avg_salary
3  FROM employees
4  GROUP BY department_id;
```

Results | Explain | Describe | Saved SQL | History

MAX_AVG_SALARY	MIN_AVG_SALARY
19333.333333333333333333333333333333333	3371.428571428571428571428571428571428571

6. Write a query that will return the average of the maximum salaries in each department for the employees table.

```
1  SELECT AVG(MAX(salary)) AS avg_max_salary
2  FROM employees
3  GROUP BY department_id;
```

Results | Explain | Describe | Saved SQL | History

AVG_MAX_SALARY
10633.333333333333333333333333333333333

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.

```
1  SELECT manager_id, job_id, SUM(salary) AS total_salary
2  FROM employees
3  GROUP BY ROLLUP(manager_id, job_id);
```

Results | Explain | Describe | Saved SQL | History

MANAGER_ID	JOB_ID	TOTAL_SALARY
-	AD_PRES	24000
-	-	24000
100	AD_VP	34000
100	MK_MAN	13000
100	SA_MAN	10500
100	ST_MAN	5800
100	-	63300
101	AC_MGR	12000
101	AD_ASST	17200
101	-	29200

2. Amend the previous query to also include a subtotal salary for each job_id regardless of the manager_id.

```
1  SELECT manager_id, job_id, SUM(salary) AS total_salary
2  FROM employees
3  GROUP BY CUBE(manager_id, job_id);
```

Results | Explain | Describe | Saved SQL | History

MANAGER_ID	JOB_ID	TOTAL_SALARY
-	-	24000
-	-	294200
-	AD_VP	34000
-	AC_MGR	12000
-	MK_MAN	13000
-	MK_REP	21500
-	SA_MAN	10500
-	SA_REP	48700
-	ST_MAN	5800
-	AD_ASST	17200

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_id

```
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));
```

Results | Explain | Describe | Saved SQL | History

DEPARTMENT_ID	MANAGER_ID	JOB_ID	TOTAL_SALARY
90	-	AD_PRES	24000
90	100	AD_VP	34000
20	100	MK_MAN	13000
80	100	SA_MAN	10500
50	100	ST_MAN	5800
110	101	AC_MGR	12000
10	101	AD_ASST	17200
60	102	IT_PROG	9000

9-3: Set Operators

1. Name the different Set operators?
    - **UNION**: Combines results of two queries, removing duplicates.
    - **UNION ALL**: Combines results of two queries, including duplicates.
    - **INTERSECT**: Returns only the rows that are common between two queries.
    - **MINUS**: 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.

```
1  SELECT employee_id, job_id, hire_date AS start_date, department_id
2  FROM employees
3  UNION
4  SELECT employee_id, job_id, start_date, department_id
5  FROM job_history;
```

Results | Explain | Describe | Saved SQL | History

EMPLOYEE_ID	JOB_ID	START_DATE	DEPARTMENT_ID
100	AD_PRES	17-Jun-2002	90
101	AC_ACCOUNT	21-Sep-1989	110
101	AC_MGR	28-Oct-1993	110
101	AD_VP	21-Sep-2004	90
102	AD_VP	13-Jan-2008	90
102	IT_PROG	13-Jan-1993	60
103	IT_PROG	03-Jan-2005	60
104	IT_PROG	21-May-2006	60

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.

```
1  SELECT employee_id, job_id, hire_date AS start_date, department_id
2  FROM employees
3  UNION ALL
4  SELECT employee_id, job_id, start_date, department_id
5  FROM job_history
6  ORDER BY employee_id;
```

Results | Explain | Describe | Saved SQL | History

EMPLOYEE_ID	JOB_ID	START_DATE	DEPARTMENT_ID
100	AD_PRES	17-Jun-2002	90
101	AC_MGR	28-Oct-1993	110
101	AD_VP	21-Sep-2004	90
101	AC_ACCOUNT	21-Sep-1989	110
102	AD_VP	13-Jan-2008	90
102	IT_PROG	13-Jan-1993	60
103	IT_PROG	03-Jan-2005	60
104	IT_PROG	21-May-2006	60

4. List all employees who have not changed jobs even once. (Such employees are not found in the job_history table)

```
SELECT e.employee_id, e.job_id, e.hire_date, e.department_id
FROM employees e
WHERE NOT EXISTS (
    SELECT 1
    FROM job_history jh
    WHERE e.employee_id = jh.employee_id);
```

Results | Explain | Describe | Saved SQL | History

EMPLOYEE_ID	JOB_ID	HIRE_DATE	DEPARTMENT_ID
100	AD_PRES	17-Jun-2002	90
103	IT_PROG	03-Jan-2005	60
104	IT_PROG	21-May-2006	60
107	IT_PROG	07-Feb-2014	60
124	ST_MAN	16-Nov-2014	50
141	ST_CLERK	17-Oct-2010	50
142	ST_CLERK	29-Jan-2012	50
143	ST_CLERK	15-Mar-2013	50

5. List the employees that HAVE changed their jobs at least once.

```
SELECT e.employee_id, e.job_id, e.hire_date, e.department_id
FROM employees e
WHERE EXISTS (
    SELECT 1
    FROM job_history jh
    WHERE e.employee_id = jh.employee_id);
```

Results | Explain | Describe | Saved SQL | History

EMPLOYEE_ID	JOB_ID	HIRE_DATE	DEPARTMENT_ID
101	AD_VP	21-Sep-2004	90
102	AD_VP	13-Jan-2008	90
176	SA_REP	24-Mar-2013	80
200	AD_ASST	17-Sep-2002	10
201	MK_MAN	17-Feb-2011	20

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, NVL(salary, 0) AS salary
FROM employees
UNION
SELECT employee_id, job_id, 0 AS salary
FROM job_history;
```

Results | Explain | Describe | Saved SQL | History

EMPLOYEE_ID	JOB_ID	SALARY
100	AD_PRES	24000
101	AC_ACCOUNT	0
101	AC_MGR	0
101	AD_VP	17000
102	AD_VP	17000
102	IT_PROG	0
103	IT_PROG	9000
104	IT_PROG	6000