Name: Danny Chen
Assignment: Database Programming: Sections 9
Date: 10/22/24

Table of Content

9.1 - Using Group By and Having Clauses

Vocab

Word	Definition
HAVING	Used to specify which groups are to be displayed; restricts groups that do not meet group criteria
GROUP BY	Divides the rows in a table into groups

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;

ANS:
The issue is that group functions cannot be used in the WHERE clause since WHERE is used only to include/exclude individual rows, not groups of rows.

```
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;

ANS:
GROUP BY is needed to ensure that the query is calculating the amount of titles that each unique cd_number has

```
SELECT cd_number, COUNT(title)
FROM d_cds
WHERE cd_number
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;

ANS:
Any columns in the SELECT clause that are not part of a group function (e.g. MAX) must be listed in the GROUP BY clause. Thus, artist must be listed in the GROUP BY clause in the query and not the Artist alias because GROUP BY is processed first before the SELECT.

```
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, artist;

d. SELECT loc_type, rental_fee AS Fee
   FROM d_venues
   WHERE id <100
   GROUP BY "Fee"
   ORDER BY 2;

ANS:
loc_type needed to be put into the GROUP BY clause because it is not part of a group function. "Fee" does not work in GROUP BY to reference rental_fee to the order of query processing. The order is as follows for the query above: FROM, WHERE, GROUP BY, SELECT, ORDER BY. The GROUP BY is before SELECT, so "Fee" has not been assigned as an alias yet.

SELECT loc_type, rental_fee AS Fee
FROM d_venues
WHERE id < 100
GROUP BY loc_type, rental_fee
ORDER BY rental_fee;

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);

ANS:
SELECT track, MAX(song_id)
FROM d_track_listings
WHERE track IN (1, 2, 3)
GROUP BY track;

TRACK	MAX(SONG_ID)
1	46
2	47
3	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.
   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.

ANS:

```
SELECT
        ROUND(MAX(AVG(salary)), 2) "Max Salary By Dept",
        ROUND(MIN(AVG(salary)), 2) "Min Salary By Dept"
FROM employees
GROUP BY department_id;
```

Max Salary By Dept	Min Salary By Dept
19333.33	3371.43

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

ANS:

```
SELECT
        ROUND(AVG(MAX(salary)), 2) "Avg max salaries in each department"
FROM employees
GROUP BY department_id;
```

Avg max salaries in each department
10633.33

9.2 - Using Rollup and Cube Operations, and Grouping Sets
Vocab

Word	Definition

ROLLUP	Used to create subtotals that roll up from the most detailed level to a grand total, following a grouping list specified in the clause
CUBE	An extension to the GROUP BY clause like ROLLUP that produces cross-tabulation reports
GROUPING SETS	Used to specify multiple groupings of data

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.

ANS:

```
SELECT manager_id, job_id, SUM(salary)
FROM employees
GROUP BY ROLLUP (manager_id, job_id)
```

MANAGER_ID	JOB_ID	SUM(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
102	IT_PROG	9000
102	-	9000
103	IT_PROG	26000
103	-	26000
124	ST_CLERK	14300
124	SR_ST_CLRK	3500
124	-	17800
149	SA_REP	48700
149	SR_SA_REP	19100
149	-	67800
201	MK_REP	21500
201	SR_MK_REP	8600
201	-	30100
205	AC_ACCOUNT	27000
205	-	27000
-	-	294200

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

ANS:

```
SELECT manager_id, job_id, SUM(salary),
   GROUPING(manager_id) AS "Manager Subtotal",
   GROUPING(job_id) AS "Job Subtotal"
FROM employees
GROUP BY CUBE (manager_id, job_id);
```

MANAGER_ID	JOB_ID	SUM(SALARY)	Manager Subtotal	Job Subtotal
-	-	24000	0	1
-	-	294200	1	1
-	AD_VP	34000	1	0
-	AC_MGR	12000	1	0
-	MK_MAN	13000	1	0
-	MK_REP	21500	1	0
-	SA_MAN	10500	1	0
-	SA_REP	48700	1	0
-	ST_MAN	5800	1	0
-	AD_ASST	17200	1	0
-	AD_PRES	24000	0	0
-	AD_PRES	24000	1	0
-	IT_PROG	35000	1	0
-	ST_CLERK	14300	1	0
-	SR_MK_REP	8600	1	0
-	SR_SA_REP	19100	1	0
-	AC_ACCOUNT	27000	1	0
-	SR_ST_CLRK	3500	1	0

100	-	63300	0	1
100	AD_VP	34000	0	0
100	MK_MAN	13000	0	0
100	SA_MAN	10500	0	0
100	ST_MAN	5800	0	0
101	-	29200	0	1
101	AC_MGR	12000	0	0
101	AD_ASST	17200	0	0
102	-	9000	0	1
102	IT_PROG	9000	0	0
103	-	26000	0	1
103	IT_PROG	26000	0	0
124	-	17800	0	1
124	ST_CLERK	14300	0	0
124	SR_ST_CLRK	3500	0	0
149	-	67800	0	1
149	SA_REP	48700	0	0
149	SR_SA_REP	19100	0	0
201	-	30100	0	1
201	MK_REP	21500	0	0
201	SR_MK_REP	8600	0	0
205	-	27000	0	1
205	AC_ACCOUNT	27000	0	0

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

ANS:

```
SELECT department_id, manager_id, job_id, SUM(salary)
FROM employees
GROUP BY GROUPING SETS
((department_id, manager_id, job_id), (manager_id, job_id),
(department_id, manager_id))
```

DEPARTMENT_ID	MANAGER_ID	JOB_ID	SUM(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
60	103	IT_PROG	26000
50	124	ST_CLERK	14300
50	124	SR_ST_CLRK	3500
-	149	SA_REP	7000
80	149	SA_REP	19600
85	149	SA_REP	22100
80	149	SR_SA_REP	9600
85	149	SR_SA_REP	9500
20	201	MK_REP	21500
20	201	SR_MK_REP	8600
110	205	AC_ACCOUNT	27000
80	100	-	10500

80	149	-	29200
85	149	-	31600
90	100	-	34000
110	101	-	12000
50	124	-	17800
20	100	-	13000
50	100	-	5800
90	-	-	24000
10	101	-	17200
20	201	-	30100
60	102	-	9000
60	103	-	26000
-	149	-	7000
110	205	-	27000
-	-	AD_PRES	24000
-	100	AD_VP	34000
-	100	MK_MAN	13000
-	100	SA_MAN	10500
-	100	ST_MAN	5800
-	101	AC_MGR	12000
-	101	AD_ASST	17200
-	102	IT_PROG	9000
-	103	IT_PROG	26000
-	124	ST_CLERK	14300
-	124	SR_ST_CLRK	3500
-	149	SA_REP	48700

-	149	SR_SA_REP	19100
-	201	MK_REP	21500
-	201	SR_MK_REP	8600
-	205	AC_ACCOUNT	27000

9.3 - Using Set Operators

Vocab

Word	Definition
UNION	operator that returns all rows from both tables and eliminates duplicates
TO_CHAR(NULL), TO_DATE(NULL), TO_NUMBER(NULL)	columns that were made up to match queries in another table that are not in both tables
UNION ALL	operator that returns all rows from both tables, including duplicates
Set operators	used to combine results into one single result from multiple SELECT statements
MINUS	operator that returns rows that are unique to each table
INTERSECT	operator that returns rows common to both tables

1. Name the different Set operators?
    a. **ANS:**
        i. UNION
        ii. UNION ALL
        iii. INTERSECT
        iv. 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.

ANS:

SELECT employee_id, job_id, hire_date, TO_DATE(NULL) start_date, department_id

```
FROM employees
   UNION
SELECT employee_id, job_id, TO_DATE(NULL), start_date, department_id
FROM job_history;
```

EMPLOYEE_ID	JOB_ID	HIRE_DATE	START_DATE	DEPARTMENT_ID
100	AD_PRES	17-Jun-02	-	90
101	AC_ACCOUNT	-	21-Sep-89	110
101	AC_MGR	-	28-Oct-93	110
101	AD_VP	21-Sep-04	-	90
102	AD_VP	13-Jan-08	-	90
102	IT_PROG	-	13-Jan-93	60
103	IT_PROG	3-Jan-05	-	60
104	IT_PROG	21-May-06	-	60
107	IT_PROG	7-Feb-14	-	60
114	ST_CLERK	-	24-Mar-98	50
122	ST_CLERK	-	1-Jan-99	50
124	ST_MAN	16-Nov-14	-	50
141	ST_CLERK	17-Oct-10	-	50
142	ST_CLERK	29-Jan-12	-	50
143	ST_CLERK	15-Mar-13	-	50
144	ST_CLERK	9-Jul-13	-	50
149	SA_MAN	29-Jan-15	-	80
174	SA_REP	11-May-11	-	80
176	SA_MAN	-	1-Jan-99	80
176	SA_REP	24-Mar-13	-	80
176	SA_REP	-	24-Mar-98	80

178	SA_REP	24-May-14	-	-
200	AC_ACCOUNT	-	1-Jul-94	90
200	AD_ASST	17-Sep-02	-	10
200	AD_ASST	-	17-Sep-87	90
201	MK_MAN	17-Feb-11	-	20
201	MK_REP	-	17-Feb-96	20
202	MK_REP	17-Aug-13	-	20
205	AC_MGR	7-Jun-09	-	110
206	AC_ACCOUNT	7-Jun-09	-	110
207	SR_SA_REP	12-Mar-09	-	85
208	SA_REP	25-Oct-09	-	85
209	SA_REP	6-Feb-11	-	85
210	SA_REP	16-Aug-12	-	85
212	SR_SA_REP	1-Sep-12	-	80
215	SR_MK_REP	2-Nov-04	-	20
216	SR_ST_CLRK	1-Apr-14	-	50
217	MK_REP	9-Feb-13	-	20
219	MK_REP	16-Dec-15	-	20
220	ST_CLERK	6-Jul-15	-	50
222	IT_PROG	29-Aug-08	-	60
223	IT_PROG	18-Nov-13	-	60
224	AC_ACCOUNT	25-Jul-07	-	110
225	AD_ASST	13-Jun-11	-	10
226	AD_ASST	17-May-15	-	10
227	AD_ASST	1-Feb-12	-	10

228	MK_REP	6-Jan-97	-	20
231	AC_ACCOUNT	11-May-09	-	110
232	AC_ACCOUNT	14-Apr-13	-	110
235	MK_REP	16-Dec-15	-	20

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.

ANS: The previous statement had 50 rows total. However, using UNION ALL did not result in extra rows although UNION excludes duplicates and UNION ALL includes them. There are still 50 rows.

```
SELECT employee_id, job_id, hire_date, TO_DATE(NULL) start_date, department_id
FROM employees
   UNION ALL
SELECT employee_id, job_id, TO_DATE(NULL), start_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)

ANS:

```
SELECT DISTINCT employee_id
FROM employees
   MINUS
SELECT DISTINCT employee_id
FROM job_history
```

EMPLOYEE_ID
100
103
104
107

124
141
142
143
144
149
174
178
202
205
206
207
208
209
210
212
215
216
217
219
220
222
223
224
225

226
227
228
231
232
235

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

ANS:

```
SELECT DISTINCT employee_id
FROM employees
   INTERSECT
SELECT DISTINCT employee_id
FROM job_history
```

EMPLOYEE_ID
101
102
176
200
201

5 rows returned in 0.00 seconds Download

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.

ANS:

```
SELECT employee_id, job_id, salary
FROM employees
   UNION
SELECT employee_id, job_id, NVL(TO_NUMBER(NULL), 0)
FROM job_history
```

ORDER BY employee_id;

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
107	IT_PROG	4200
114	ST_CLERK	0
122	ST_CLERK	0
124	ST_MAN	5800
141	ST_CLERK	3500
142	ST_CLERK	3100
143	ST_CLERK	2600
144	ST_CLERK	2500
149	SA_MAN	10500
174	SA_REP	11000
176	SA_MAN	0
176	SA_REP	0
176	SA_REP	8600
178	SA_REP	7000
200	AC_ACCOUNT	0

200	AD_ASST	0
200	AD_ASST	4400
201	MK_MAN	13000
201	MK_REP	0
202	MK_REP	3900
205	AC_MGR	12000
206	AC_ACCOUNT	8300
207	SR_SA_REP	9500
208	SA_REP	7500
209	SA_REP	7300
210	SA_REP	7300
212	SR_SA_REP	9600
215	SR_MK_REP	8600
216	SR_ST_CLRK	3500
217	MK_REP	4000
219	MK_REP	3700
220	ST_CLERK	2600
222	IT_PROG	8000
223	IT_PROG	7800
224	AC_ACCOUNT	8100
225	AD_ASST	4300
226	AD_ASST	4100
227	AD_ASST	4400
228	MK_REP	5000
231	AC_ACCOUNT	5400

232	AC_ACCOUNT	5200
235	MK_REP	4900