**1. Display the names of the employees whose salary is the same as the lowest salaried employee in any department.**

SELECT first\_name || ' ' || last\_name "Name"

FROM employees

WHERE salary = ANY

(SELECT MIN(salary)

FROM employees

GROUP BY department\_id);

Name

-------------------

Neena Kochhar

Lex De Haan

Bruce Ernst

Diana Lorentz

Randall Matos

Peter Vargas

Jonathon Taylor

Kimberely Grant

Jennifer Whalen

Pat Fay

William Gietz

**2. Display the names of the employee(s) whose salary is the lowest in each department.**

SELECT first\_name || ' ' || last\_name "Name"

FROM employees

WHERE (salary, department\_id) IN

(SELECT MIN(salary),department\_id

FROM employees

GROUP BY department\_id);

Name

--------------------------

Neena Kochhar

Lex De Haan

Pat Fay

William Gietz

Randall Matos

Peter Vargas

Jonathon Taylor

Diana Lorentz

Jennifer Whalen

**3. Give each of the employees in question 2 a $100 bonus.**

UPDATE employees

SET salary = salary + 100

WHERE first\_name IN

(SELECT first\_name

FROM employees

WHERE (salary, department\_id) IN

(SELECT MIN(salary), department\_id

FROM employees

GROUP BY department\_id));

Name SALARY

---------------------------------- ----------

Pat Fay                                          6100

Neena Kochhar                               17100

Lex De Haan                                   17100

William Gietz                               8400

Randall Matos                                2600

Peter Vargas                                  2600

Jonathon Taylor                             8700

Jennifer Whalen                             4500

Diana Lorentz                                4300

**4. Create a view named ALLEMPS that consists of all employees includes employee\_id, last\_name, salary, department\_id, department\_name, city and country (if applicable)**

CREATE VIEW ALLEMPS AS

SELECT employee\_id, last\_name, salary, department\_id, department\_name, city, country\_id

FROM employees

LEFT JOIN departments USING(department\_id)

LEFT JOIN locations USING(location\_id);

EMPLOYEE\_ID LAST\_NAME SALARY DEPART\_ID DEPART\_NAME CITY COUNTRY\_ID

------------ --------- ------ --------- ------------ ----------- ------------

103 Hunold 9000 60 IT Southlake US

104 Ernst 6000 60 IT Southlake US

107 Lorentz 4400 60 IT Southlake US

124 Mourgos 5800 50 Shipping South San Francisco US

141 Rajs 3500 50 Shipping South San Francisco US

142 Davies 3100 50 Shipping South San Francisco US

143 Matos 2700 50 Shipping South San Francisco US

205 Higgins 12000 110 Accounting Seattle US

206 Gietz 8500 110 Accounting Seattle US

207 Lee 90 Executive Seattle US

100 King 24000 90 Executive Seattle US

101 Kochhar 17200 90 Executive Seattle US

102 De Haan 17200 90 Executive Seattle US

200 Whalen 2700 10 Admin Seattle US

201 Hartstein 13000 20 Marketing Toronto CA

202 Fay 6200 20 Marketing Toronto CA

149 Zlotkey 10500 80 Sales Oxford UK

174 Abel 11000 80 Sales Oxford UK

176 Taylor 8800 80 Sales Oxford UK

178 Grant 7000

**5. Use the ALLEMPS view to:**

**a.Display the employee\_id, last\_name, salary and city for all employees**

SELECT employee\_id, last\_name, salary, city

FROM ALLEMPS;

Output:

EMPLOYEE\_ID LAST\_NAME SALARY CITY

----------- ------------------------- ---------- ---------------

103 Hunold 9000 Southlake

104 Ernst 6000 Southlake

107 Lorentz 4300 Southlake

124 Mourgos 5800 South San Francisco

141 Rajs 3500 South San Francisco

142 Davies 3100 South San Francisco

143 Matos 2600 South San Francisco

144 Vargas 2600 South San Francisco

205 Higgins 12000 Seattle

206 Gietz 8400 Seattle

207 Lee Seattle

100 King 24000 Seattle

101 Kochhar 17100 Seattle

102 De Haan 17100 Seattle

200 Whalen 2600 Seattle

201 Hartstein 13000 Toronto

202 Fay 6100 Toronto

149 Zlotkey 10500 Oxford

174 Abel 11000 Oxford

176 Taylor 8700 Oxford

178 Grant 7000

**b.Display the total salary of all employees by city**

SELECT SUM(salary) "Total Salary", city

FROM ALLEMPS

GROUP BY city

HAVING city IS NOT NULL;

Total Salary CITY

------------ ------------------------------

81200 Seattle

17600 South San Francisco

19100 Toronto

19300 Southlake

30200 Oxford

**c. Increase the salary of the lowest paid employee(s) in each department by 100**

UPDATE ALLEMPS

SET salary = salary + 100

WHERE last\_name IN

(SELECT last\_name

FROM ALLEMPS

WHERE (salary, department\_id) IN

(SELECT MIN(salary), department\_id

FROM ALLEMPS

GROUP BY department\_id));

**d. What happens if you try to insert an employee by providing values for all columns in this view?**

SQL Error -> ORA-01776: cannot modify more than one base table through a join view

**5. Delete the employee named Vargas. Did it work? Show proof.**

DELETE FROM ALLEMPS

WHERE last\_name = 'Vargas';

LAST\_NAME

-------------------------

Abel

Davies

De Haan

Ernst

Fay

Gietz

Grant

Hartstein

Higgins

Hunold

King

Kochhar

Lee

Lorentz

Matos

Mourgos

Rajs

Taylor

Whalen

Zlotkey

**6. Create a view named ALLDEPTS that consists of all departments and includes department\_id, department\_name, city and country (if applicable)**

CREATE VIEW ALLDEPTS AS

SELECT department\_id, department\_name, city, country\_id

FROM departments

LEFT JOIN locations USING(location\_id);

**7. Use the ALLDEPTS view to:**

**a. For all departments display the department\_id, name and city**

SELECT department\_id, department\_name, city

FROM ALLDEPTS;

DEPARTMENT\_ID DEPARTMENT\_NAME CITY

------------- ------------------------------ -------------------

60 IT Southlake

50 Shipping South San Francisco

190 Contracting Seattle

110 Accounting Seattle

90 Executive Seattle

10 Administration Seattle

20 Marketing Toronto

80 Sales Oxford

**b. For each city that has departments located in it, display the number of departments by city**

SELECT COUNT(department\_id) "# of departments by city", city

FROM ALLDEPTS

GROUP BY city;

# of departments by city CITY

------------------------ ------------------------------

4 Seattle

1 South San Francisco

1 Toronto

1 Southlake

1 Oxford

**8. Create a view called ALLDEPTSUMM that consists of all departments and includes for each department: department\_id, department\_name, number of employees, number of salaried employees, total salary of all employees. Number of Salaried must be different from number of employees. The difference is some get commission.**

**Important question to ask.**

**Are the “number of employees” the same as “number of salaried employees”?**

Selected Answer:

SQL:

CREATE VIEW ALLDEPTSUMM AS

SELECT d.department\_id, d.department\_name,

COUNT(e.employee\_id) "# of emp",

COUNT(f.employee\_id) - COUNT(commission\_pct) "# of

salaried emps",

SUM(f.SALARY) "TOTAL"

FROM departments d JOIN employees e

ON d.department\_id = e.department\_id

JOIN employees f

ON e.employee\_id = f.employee\_id

WHERE f.employee\_id IN

(SELECT employee\_id

FROM employees

WHERE commission\_pct IS NULL)

GROUP BY d.department\_id, d.department\_name;

**9. Use the ALLDEPTSUMM view to display department name and number of employees for departments that have more than the average number of employees**

SELECT department\_name, "# of employees"

FROM ALLDEPTSUMM

WHERE "# of employees" >

(SELECT AVG("# of employees")

FROM ALLDEPTSUMM);

DEPARTMENT\_NAME # of employees

------------------------------ ----------------------

Shipping 4

Executive 3

IT 3

**10. Use the GRANT statement to allow another student (Neptune account ) to retrieve data for your employees table and to allow them to retrieve, insert and update data in your departments table. Show proof**

GRANT SELECT ON employees TO dbs301\_163c23;

GRANT SELECT, INSERT, UPDATE ON departments TO dbs301\_163c23;

**11. Use the REVOKE statement to remove permission for that student to insert and update data in your departments table**

REVOKE INSERT, UPDATE ON departments FROM dbs301\_163c23;