**Homework Assignment 1**

*(due through Blackboard before class on 2/13/2019)*

Things you should please do, lest you should lose my tender affection:

* Edit this document, adding your solution code for each problem.
* Replace YourLastName with your last name in the name of the file before you upload it.
* Make sure your name is inside the document itself, in the **Name:** area.
* All code must be text inside of the document that I can copy and paste – do not include screenshots of your code.
* Make sure that your work is original – **do not copy your code from or share your code with anybody else that is working on these problems**. This would be a terrible idea, and lots of people seem to do it despite warnings like this. To be clear, adapting code you find on the Internet or in the lecture slides is perfectly acceptable. You can talk about the assignment with others, but don’t take code developed for this assignment from anybody else, and don’t give code developed for this assignment to anybody else. Please consider yourself warned.

**Prathima Mateti:**

**1. The Raise Procedure**

Create a copy of the HR.EMPLOYEES table from Oracle XE by running the following code:

CREATE TABLE MyEmployees AS SELECT \* FROM HR.Employees;

Now, create a new table that you will use for logging.

CREATE TABLE HW1Log

(

Message VARCHAR2(200),

TStamp TIMESTAMP DEFAULT SYSTIMESTAMP

);

Notice that the TStamp column was created with a default value. You can add a log message to the table like this:

INSERT INTO HW1Log (Message) VALUES ('Some message');

If you look at the table now, you’ll see that the timestamp column has been populated automatically. Later on, you’ll be asked to use code like this to write messages to your log table.

Now, create a stored procedure called RaiseSalary that accepts two parameters – an employee id and an integer raise percentage. The procedure will raise the salary of the specified employee by the specified percentage. It will do this by updating the Salary column in MyEmployees. Calling the procedure with an argument of 101 for the employee id and 20 for the raise percentage would raise that employee’s salary by 20 percent.

Your procedure should include exception handling. It should have a generic error handler which prints out the SQLERRM. In addition, it should handle the specific case of an employee id parameter that doesn’t match any existing employee. When that happens, you should raise a custom exception. In the handler for that custom exception, you should print out “*There is no employee with id <parameter value>*” instead of the SQLERRM. Of course, instead of <parameter value>, your error message should include the value of the employee id parameter that was passed into the procedure.

There are several ways to detect that the procedure was given the id of a non-existent employee. One is to use a property called SQL%ROWCOUNT that we didn’t discuss in class. You can find an example of how you might do that here: <https://stackoverflow.com/questions/861983/number-of-rows-affected-by-an-update-in-pl-sql>. If you choose to do it another way, that’s fine.

Every time your procedure successfully raises the salary of an employee, you should write a message to your HW1Log table in the following form:

*Employee <employee id> was raised by <raise percentage> percent.*

Make sure that you use %TYPE where applicable. Also, do a COMMIT inside your procedure (after all the other code) to make your changes public and permanent. It’s not always a good idea to have a COMMIT inside your procedures, but for the sake of this exercise, please put one in there.

Show the code for your procedure, as well as the **code and output** from the following exercises:

create or replace procedure RaiseSalary(

p\_empid MyEmployees.employee\_id%TYPE, p\_raise\_percent MyEmployees.salary%TYPE)

as

hike number(10);

count1 varchar2(10);

v\_salary varchar2(10);

begin

select salary into v\_salary from MyEmployees where EMPLOYEE\_ID=p\_empid;

hike:= v\_salary \* p\_raise\_percent/100;

v\_salary:=v\_salary+hike;

update MyEmployees set salary = v\_salary where EMPLOYEE\_ID = p\_empid;

insert into HW1log1(Message) values('Employee was raised by'|| p\_raise\_percent);

commit;

EXCEPTION

when no\_data\_found then

dbms\_output.put\_line('There is no employee with ID'||p\_empid);

insert into HW1log1(Message) values('There is no employee with id '||p\_empid );

when others then

dbms\_output.put\_line(SQLERRM);

end RaiseSalary;

* Show the initial salary of employee 179 (can do this with a standard SQL query)

select employee\_id,first\_name,salary from myemployees where employee\_id=179;

179 Charles 6200

* Call your procedure with an employee id of 179 and a percentage of 10

exec RaiseSalary(179,10);

PL/SQL procedure successfully completed.

* Show the salary of employee 179 after the procedure call (can do this with a standard SQL query)

select employee\_id,first\_name,salary from myemployees where employee\_id=179;

179 Charles 6820

* Show the initial salary of employee 202 (can do this with a standard SQL query)

select employee\_id,first\_name,salary from myemployees where employee\_id=202;

202 Pat 6000

* Call your procedure with an employee id of 202 and a percentage of 15

exec RaiseSalary(202,15);

PL/SQL procedure successfully completed.

* Show the salary of employee 202 after the procedure call (can do this with a standard SQL query)

select employee\_id,first\_name,salary from myemployees where employee\_id=202;

202 Pat 6900

* Call your procedure with an employee id of 500 and a percentage of 20

exec RaiseSalary(500,20);

PL/SQL procedure successfully completed.

* Show all of the data from your HW1Log table (can do this with a standard SQL query) – order your results by ascending TStamp value.

select \* from hw1log1;

Employee was raised by10 13-FEB-19 09.28.32.948072000 PM

Employee was raised by15 13-FEB-19 09.29.53.237074000 PM

There is no employee with id500 13-FEB-19 09.36.08.848759000 PM

There is no employee with id 500 13-FEB-19 09.38.38.403708000 PM

**2. But Money Doesn’t Grow on Trees**

The company had a good year, and so we want to give out some raises. We generally think that a good raise is 5% of a person’s salary. However, we don’t have an unlimited amount of money. There are only $25,000 dollars available in our raise budget, and that isn’t enough to give everybody a 5% raise, so we need to make some choices.

Management has decided to give people 5% raises until the money runs out. Since there isn’t enough money available for everybody, we need a way to prioritize who gets the raises. Management determines that raises should be prioritized in salary order – the employees with the lowest salaries should get raises first. If two people have the same salary, they should be prioritized by their hire date, with employees who were hired first having higher priority.

Write a procedure called AssignRaises that determines who will get raises. Your procedure should take the raise percentage (5% in our example above) and raise budget ($25,000 in our example above) as parameters, so that we can vary them when we need to.

The procedure should use a cursor to list the employees in the priority order described above. (Remember that you can use the ORDER BY clause in SQL to do this.) It will loop through the cursor and give raises to the employees using the RaiseSalary procedure you created in part 1. However, it needs to keep track of the declining budget, and not give more than $25,000 in raises. At a certain point, the remaining budget will have declined to the point that there is not enough money left to give the next person in the priority order a full 5% raise. When that happens, the procedure should stop giving raises at all, even if there is some money left in the raise budget.

For each person who does not get a raise, the procedure should write a message to the HW1Log table in the following form:

*Not enough money left to give a raise to employee <employee id>.*

Your procedure should finish by printing out the following report.

*Number of employees who received raises: <put value here>*

*Number of employees who did not receive raises: <put value here>*

*Amount of money left unused in the raise budget: <put value here>*

Again, your procedure should COMMIT its changes at the end, just for the sake of this exercise.

Show the code for your procedure, as well as the **code and output** from the following exercises:

create or replace procedure RaiseAssign(

p\_budget MyEmployees.employee\_id%TYPE, p\_raise\_percent MyEmployees.salary%TYPE)

as

v\_budget varchar2(10);

v\_count number(10);

rem\_count number (10);

total\_count number(10);

hike number(10);

v\_employee\_id number(10);

v\_salary varchar2(10);

CURSOR c1 IS select employee\_id,salary from myemployees order by salary,hire\_date asc;

begin

v\_budget:=p\_budget;

v\_count:=0;

FOR item IN c1

LOOP

hike:=item.salary \* p\_raise\_percent/100;

IF hike < v\_budget THEN

RaiseSalary(item.employee\_id,p\_raise\_percent);

v\_budget:= v\_budget - hike;

v\_count:=v\_count + 1;

else

insert into HW1Log3(Message) values('Not enough money left to give a raise to employee'|| item.employee\_id);

END IF;

END LOOP;

commit;

select count (\*) into total\_count from myemployees;

rem\_count:= total\_count - v\_count;

dbms\_output.put\_line('Number of employees who received raises: '||v\_count);

dbms\_output.put\_line('Number of employees who did not receive raises: '||rem\_count);

dbms\_output.put\_line('Amount of money left unused in the raise budget: '||v\_budget);

END RaiseAssign;

* Drop the MyEmployees table (DROP TABLE MyEmployees) and recreate it using the code from part 1 (CREATE TABLE MyEmployees AS SELECT \* FROM HR.Employees)

drop table myemployees;

Table MYEMPLOYEES dropped.

create table myemployees as select \* from hr.employees;

Table MYEMPLOYEES created.

* Truncate the HW1Log table (TRUNCATE TABLE HW1Log) to remove prior records.

Error starting at line : 49 in command -

TRUNCATE TABLE HW1Log

Error report -

ORA-00054: resource busy and acquire with NOWAIT specified or timeout expired

00054. 00000 - "resource busy and acquire with NOWAIT specified or timeout expired"

\*Cause: Interested resource is busy.

\*Action: Retry if necessary or increase timeout.

So creating other hw1log3;

Table HW1LOG3 created.

* Run the AssignRaises procedure with a raise percentage of **5** and a raise budget of **25000**.

exec RaiseAssign(25000,5);

Number of employees who received raises: 92

Number of employees who did not receive raises: 15

Amount of money left unused in the raise budget: 430

PL/SQL procedure successfully completed.

* Show all of the data from your HW1Log table (can do this with a standard SQL query) – order your results by ascending TStamp value.

select \* from HW1log3;

Not enough money left to give a raise to employee162 13-FEB-19 10.01.50.590088000 PM

Not enough money left to give a raise to employee149 13-FEB-19 10.01.50.590221000 PM

Not enough money left to give a raise to employee114 13-FEB-19 10.01.50.590252000 PM

Not enough money left to give a raise to employee174 13-FEB-19 10.01.50.590277000 PM

Not enough money left to give a raise to employee148 13-FEB-19 10.01.50.590301000 PM

Not enough money left to give a raise to employee168 13-FEB-19 10.01.50.590362000 PM

Not enough money left to give a raise to employee147 13-FEB-19 10.01.50.590397000 PM

Not enough money left to give a raise to employee205 13-FEB-19 10.01.50.590422000 PM

Not enough money left to give a raise to employee108 13-FEB-19 10.01.50.590487000 PM

Not enough money left to give a raise to employee201 13-FEB-19 10.01.50.590513000 PM

Not enough money left to give a raise to employee146 13-FEB-19 10.01.50.590537000 PM

Not enough money left to give a raise to employee145 13-FEB-19 10.01.50.590561000 PM

Not enough money left to give a raise to employee102 13-FEB-19 10.01.50.590585000 PM

Not enough money left to give a raise to employee101 13-FEB-19 10.01.50.590608000 PM

Not enough money left to give a raise to employee100 13-FEB-19 10.01.50.590632000 PM

* Show the employee id and salary of all rows from the MyEmployees table (can do this with a standard SQL query) – order your results by employee id.

100 24000

101 17000

102 17000

103 9450

104 6300

105 5040

106 5040

107 4410

108 12008

109 9450

110 8610

111 8085

112 8190

113 7245

114 11000

115 3255

116 3045

117 2940

118 2730

119 2625

120 8400

121 8610

122 8295

123 6825

124 6090

125 3360

126 2835

127 2520

128 2310

129 3465

130 2940

131 2625

132 2205

133 3465

134 3045

135 2520

136 2310

137 3780

138 3360

139 2835

140 2625

141 3675

142 3255

143 2730

144 2625

145 14000

146 13500

147 12000

148 11000

149 10500

150 10500

151 9975

152 9450

153 8400

154 7875

155 7350

156 10500

157 9975

158 9450

159 8400

160 7875

161 7350

162 10500

163 9975

164 7560

165 7140

166 6720

167 6510

168 11500

169 10500

170 10080

171 7770

172 7665

173 6405

174 11000

175 9240

176 9030

177 8820

178 7350

179 6510

180 3360

181 3255

182 2625

183 2940

184 4410

185 4305

186 3570

187 3150

188 3990

189 3780

190 3045

191 2625

192 4200

193 4095

194 3360

195 2940

196 3255

197 3150

198 2730

199 2730

200 4620

201 13000

202 6300

203 6825

204 10500

205 12008

206 8715

* Drop the MyEmployees table (DROP TABLE MyEmployees) and recreate it using the code from part 1 (CREATE TABLE MyEmployees AS SELECT \* FROM HR.Employees)

drop table myemployees;

Table MYEMPLOYEES dropped.

create table myemployees as select \* from hr.employees;

Table MYEMPLOYEES created.

* Truncate the HW1Log table (TRUNCATE TABLE HW1Log) to remove prior records.

TRUNCATE TABLE HW1Log3;

Table HW1LOG3 truncated.

* Run the AssignRaises procedure with a raise percentage of **4** and a raise budget of **26000**.

exec RaiseAssign(26000,4);

Number of employees who received raises: 104

Number of employees who did not receive raises: 3

Amount of money left unused in the raise budget: 664

PL/SQL procedure successfully completed.

* Show all of the data from your HW1Log table (can do this with a standard SQL query) – order your results by ascending TStamp value.

select \* from HW1log3;

Not enough money left to give a raise to employee102 13-FEB-19 10.07.29.365405000 PM

Not enough money left to give a raise to employee101 13-FEB-19 10.07.29.365536000 PM

Not enough money left to give a raise to employee100 13-FEB-19 10.07.29.365566000 PM

* Show the employee id and salary of all rows from the MyEmployees table (can do this with a standard SQL query) – order your results by employee id.

select employee\_id,salary from myemployees order by employee\_id asc;

100 24000

101 17000

102 17000

103 9360

104 6240

105 4992

106 4992

107 4368

108 12488

109 9360

110 8528

111 8008

112 8112

113 7176

114 11440

115 3224

116 3016

117 2912

118 2704

119 2600

120 8320

121 8528

122 8216

123 6760

124 6032

125 3328

126 2808

127 2496

128 2288

129 3432

130 2912

131 2600

132 2184

133 3432

134 3016

135 2496

136 2288

137 3744

138 3328

139 2808

140 2600

141 3640

142 3224

143 2704

144 2600

145 14560

146 14040

147 12480

148 11440

149 10920

150 10400

151 9880

152 9360

153 8320

154 7800

155 7280

156 10400

157 9880

158 9360

159 8320

160 7800

161 7280

162 10920

163 9880

164 7488

165 7072

166 6656

167 6448

168 11960

169 10400

170 9984

171 7696

172 7592

173 6344

174 11440

175 9152

176 8944

177 8736

178 7280

179 6448

180 3328

181 3224

182 2600

183 2912

184 4368

185 4264

186 3536

187 3120

188 3952

189 3744

190 3016

191 2600

192 4160

193 4056

194 3328

195 2912

196 3224

197 3120

198 2704

199 2704

200 4576

201 13520

202 6240

203 6760

204 10400

205 12488

206 8632

**3. Raises Are a Trigger for Me**

Again, **drop the MyEmployees table** (DROP TABLE MyEmployees) and **recreate it** using the code from part 1 (CREATE TABLE MyEmployees AS SELECT \* FROM HR.Employees). If you don’t do this, your results for this part won’t come out well. Also, make sure you do it before you write the trigger that will be described next.

Management has now decided that no employee salary should ever be increased by more than $400 at a time, regardless of the percentage raise or the amount of money available. You have offered to change your RaiseSalary procedure and AssignRaises procedures to accommodate this limitation, but they are concerned about the possibility of rogue applications in the company bypassing your code and updating the tables with direct SQL. They want you to make sure that no single update can raise the salary by more than $400, regardless of which application it is requesting the change.

Implement a trigger called RaiseGuard which fires on any UPDATE to the Salary column of MyEmployees and which assures that raises of more than $400 are converted to exactly $400. Remember that a BEFORE trigger can modify the values of the :new row – there is a similar example in the slides that talks about using an Oracle sequence to assign a primary key value for new records.

Every time your trigger modifies a salary value, it should write a message to the HW1Log table in the following form:

*Salary update for employee <employee id> modified from <old salary> to limit of <new salary>.*

That new salary will be $400 in this example, but use a variable to represent the max value so you can easily maintain your code whenever the value changes.

Show the code for your trigger, as well as the **code and output** from the following exercises:

CREATE OR REPLACE TRIGGER RaiseGuard

BEFORE UPDATE of salary ON MyEmployees

FOR EACH ROW

declare

max\_salary varchar2(20);

hike number(20);

emp\_id varchar2(10);

v\_salary varchar2(20);

begin

emp\_id:=:new.employee\_id;

max\_salary:=400;

hike:=:new.salary - :old.salary;

if (hike > max\_salary) then

v\_salary:= :old.salary + max\_salary;

dbms\_output.put\_line('updated salary : ' ||v\_salary);

insert into HW1Log3(Message) values('Salary update for employee'||emp\_id || 'modified from ' ||:new.salary||'to limit of'||max\_salary);

end if;

end;

* Truncate the HW1Log table (TRUNCATE TABLE HW1Log) to remove prior records.

TRUNCATE TABLE HW1Log3;

Table HW1LOG3 truncated.

* Run the AssignRaises procedure with a raise percentage of **5** and a raise budget of **25000**.

exec RaiseAssign(25000,5);

updated salary : 8600

updated salary : 8600

updated salary : 8700

updated salary : 8800

updated salary : 9000

updated salary : 9200

updated salary : 9400

updated salary : 9400

updated salary : 9400

updated salary : 9400

updated salary : 9900

updated salary : 9900

updated salary : 9900

updated salary : 10000

updated salary : 10400

updated salary : 10400

updated salary : 10400

updated salary : 10400

Number of employees who received raises: 92

Number of employees who did not receive raises: 15

Amount of money left unused in the raise budget: 430

PL/SQL procedure successfully completed.

* Show all of the data from your HW1Log table (can do this with a standard SQL query) – order your results by ascending TStamp value.

select \* from HW1Log3;

Salary update for employee121modified from 8610to limit of400 13-FEB-19 11.14.34.369182000 PM

Salary update for employee110modified from 8610to limit of400 13-FEB-19 11.14.34.370134000 PM

Salary update for employee206modified from 8715to limit of400 13-FEB-19 11.14.34.371491000 PM

Salary update for employee177modified from 8820to limit of400 13-FEB-19 11.14.34.375645000 PM

Salary update for employee176modified from 9030to limit of400 13-FEB-19 11.14.34.378131000 PM

Salary update for employee175modified from 9240to limit of400 13-FEB-19 11.14.34.381121000 PM

Salary update for employee109modified from 9450to limit of400 13-FEB-19 11.14.34.383746000 PM

Salary update for employee158modified from 9450to limit of400 13-FEB-19 11.14.34.388777000 PM

Salary update for employee152modified from 9450to limit of400 13-FEB-19 11.14.34.390122000 PM

Salary update for employee103modified from 9450to limit of400 13-FEB-19 11.14.34.390765000 PM

Salary update for employee157modified from 9975to limit of400 13-FEB-19 11.14.34.391381000 PM

Salary update for employee151modified from 9975to limit of400 13-FEB-19 11.14.34.391949000 PM

Salary update for employee163modified from 9975to limit of400 13-FEB-19 11.14.34.392613000 PM

Salary update for employee170modified from 10080to limit of400 13-FEB-19 11.14.34.394544000 PM

Salary update for employee204modified from 10500to limit of400 13-FEB-19 11.14.34.396506000 PM

Salary update for employee156modified from 10500to limit of400 13-FEB-19 11.14.34.398022000 PM

Salary update for employee150modified from 10500to limit of400 13-FEB-19 11.14.34.398663000 PM

Salary update for employee169modified from 10500to limit of400 13-FEB-19 11.14.34.399281000 PM

Not enough money left to give a raise to employee162 13-FEB-19 11.14.34.400136000 PM

Not enough money left to give a raise to employee149 13-FEB-19 11.14.34.400281000 PM

Not enough money left to give a raise to employee114 13-FEB-19 11.14.34.400320000 PM

Not enough money left to give a raise to employee174 13-FEB-19 11.14.34.400346000 PM

Not enough money left to give a raise to employee148 13-FEB-19 11.14.34.400370000 PM

Not enough money left to give a raise to employee168 13-FEB-19 11.14.34.400393000 PM

Not enough money left to give a raise to employee147 13-FEB-19 11.14.34.400417000 PM

Not enough money left to give a raise to employee205 13-FEB-19 11.14.34.400442000 PM

Not enough money left to give a raise to employee108 13-FEB-19 11.14.34.400505000 PM

Not enough money left to give a raise to employee201 13-FEB-19 11.14.34.400531000 PM

Not enough money left to give a raise to employee146 13-FEB-19 11.14.34.400555000 PM

Not enough money left to give a raise to employee145 13-FEB-19 11.14.34.400579000 PM

Not enough money left to give a raise to employee102 13-FEB-19 11.14.34.400603000 PM

Not enough money left to give a raise to employee101 13-FEB-19 11.14.34.400627000 PM

Not enough money left to give a raise to employee100 13-FEB-19 11.14.34.400651000 PM

* Show the employee id and salary of all rows from the MyEmployees table (can do this with a standard SQL query) – order your results by employee id.

select employee\_id,salary from myemployees order by employee\_id asc;

100 24000

101 17000

102 17000

103 9450

104 6300

105 5040

106 5040

107 4410

108 12008

109 9450

110 8610

111 8085

112 8190

113 7245

114 11000

115 3255

116 3045

117 2940

118 2730

119 2625

120 8400

121 8610

122 8295

123 6825

124 6090

125 3360

126 2835

127 2520

128 2310

129 3465

130 2940

131 2625

132 2205

133 3465

134 3045

135 2520

136 2310

137 3780

138 3360

139 2835

140 2625

141 3675

142 3255

143 2730

144 2625

145 14000

146 13500

147 12000

148 11000

149 10500

150 10500

151 9975

152 9450

153 8400

154 7875

155 7350

156 10500

157 9975

158 9450

159 8400

160 7875

161 7350

162 10500

163 9975

164 7560

165 7140

166 6720

167 6510

168 11500

169 10500

170 10080

171 7770

172 7665

173 6405

174 11000

175 9240

176 9030

177 8820

178 7350

179 6510

180 3360

181 3255

182 2625

183 2940

184 4410

185 4305

186 3570

187 3150

188 3990

189 3780

190 3045

191 2625

192 4200

193 4095

194 3360

195 2940

196 3255

197 3150

198 2730

199 2730

200 4620

201 13000

202 6300

203 6825

204 10500

205 12008

206 8715

Depending on how you have implemented the AssignRaises procedure, this new trigger may invalidate its calculations of how much money is left in the raise budget. However, you can ignore that for now, and you don’t need to update AssignRaises to deal with the impacts of the trigger.