**1. Functional Programming (Ha!)**

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

Management is still investigating the issue of raises. They’re considering a new approach to raises where employees who make more than average salary get a 3% raise and employees who make an average or less than average salary get a 5% raise.

Create a PL/SQL function called RaiseCalculator that will implement the above rule. The function will take a current salary as an argument and return the new raised salary (not the amount of the raise). Note that the function should continue to work correctly even after the employees and salaries you currently find in the table change in the future, so don’t hard-code any literal values in your function.

SELECT employee\_id, salary, RaiseCalculator(salary) AS raised\_salary

FROM myemployees

ORDER BY salary;

CREATE OR REPLACE FUNCTION

RaiseCalculator(emp\_salary MyEmployees.salary%TYPE) RETURN NUMBER

AS

f\_avg\_salary NUMBER :=0; --avg emp salary

f\_below\_avg NUMBER := 5; --raise % for sal <= avg\_sal

f\_above\_avg NUMBER :=3; --raise % for sal > avg\_sal

f\_raised\_salary NUMBER:=0; --total raised salary after raise

BEGIN

SELECT AVG(e.salary) INTO f\_avg\_salary FROM MyEmployees e;--get avg emp salary

IF emp\_salary <= f\_avg\_salary THEN

f\_raised\_salary := emp\_salary\*f\_below\_avg/100;

ELSE f\_raised\_salary := emp\_salary\*f\_above\_avg/100;

END IF;

f\_raised\_salary:=f\_raised\_salary+ emp\_salary;

RETURN f\_raised\_salary;

END RaiseCalculator;

SELECT AVG(e.salary) FROM MyEmployees e;

SELECT employee\_id, salary, RaiseCalculator(salary) AS raised\_salary

FROM myemployees

ORDER BY salary;

Result Set:

**132 2100 2205**

**128 2200 2310**

**136 2200 2310**

**127 2400 2520**

**135 2400 2520**

**119 2500 2625**

**131 2500 2625**

**140 2500 2625**

**144 2500 2625**

**182 2500 2625**

**191 2500 2625**

**118 2600 2730**

**143 2600 2730**

**198 2600 2730**

**199 2600 2730**

**126 2700 2835**

**139 2700 2835**

**117 2800 2940**

**130 2800 2940**

**183 2800 2940**

**195 2800 2940**

**116 2900 3045**

**134 2900 3045**

**190 2900 3045**

**187 3000 3150**

**197 3000 3150**

**115 3100 3255**

**142 3100 3255**

**181 3100 3255**

**196 3100 3255**

**125 3200 3360**

**138 3200 3360**

**180 3200 3360**

**194 3200 3360**

**129 3300 3465**

**133 3300 3465**

**186 3400 3570**

**141 3500 3675**

**137 3600 3780**

**189 3600 3780**

**188 3800 3990**

**193 3900 4095**

**192 4000 4200**

**185 4100 4305**

**107 4200 4410**

**184 4200 4410**

**200 4400 4620**

**105 4800 5040**

**106 4800 5040**

**124 5800 6090**

**104 6000 6300**

**202 6000 6300**

**173 6100 6405**

**167 6200 6510**

**179 6200 6510**

**166 6400 6720**

**123 6500 6695**

**203 6500 6695**

**165 6800 7004**

**113 6900 7107**

**155 7000 7210**

**161 7000 7210**

**178 7000 7210**

**164 7200 7416**

**172 7300 7519**

**171 7400 7622**

**154 7500 7725**

**160 7500 7725**

**111 7700 7931**

**112 7800 8034**

**122 7900 8137**

**120 8000 8240**

**153 8000 8240**

**159 8000 8240**

**110 8200 8446**

**121 8200 8446**

**206 8300 8549**

**177 8400 8652**

**176 8600 8858**

**175 8800 9064**

**103 9000 9270**

**109 9000 9270**

**152 9000 9270**

**158 9000 9270**

**157 9500 9785**

**163 9500 9785**

**151 9500 9785**

**170 9600 9888**

**204 10000 10300**

**169 10000 10300**

**156 10000 10300**

**150 10000 10300**

**162 10500 10815**

**149 10500 10815**

**148 11000 11330**

**114 11000 11330**

**174 11000 11330**

**168 11500 11845**

**147 12000 12360**

**205 12008 12368.24**

**108 12008 12368.24**

**201 13000 13390**

**146 13500 13905**

**145 14000 14420**

**102 17000 17510**

**101 17000 17510**

**100 24000 24720**

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

Create a new table that you will use for logging.

CREATE TABLE HW2Log

(

Message VARCHAR2(200),

TStamp TIMESTAMP DEFAULT SYSTIMESTAMP

);

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.

Every time your trigger modifies a salary value, it should write a message to the HW2Log table in the following form:

*Salary update for employee <employee id> modified from <what salary would have been without trigger> to limit of <what salary is now after effects of trigger>.*

This new trigger will likely invalidate the calculations of your AssignRaises procedure concerning 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.

CREATE OR REPLACE TRIGGER RaiseGuard

AFTER UPDATE ON MyEmployees

FOR EACH ROW

DECLARE

t\_sal\_diff NUMBER:=0;

BEGIN

t\_sal\_diff := :new.Salary - :old.Salary;

IF t\_sal\_diff > 400 THEN

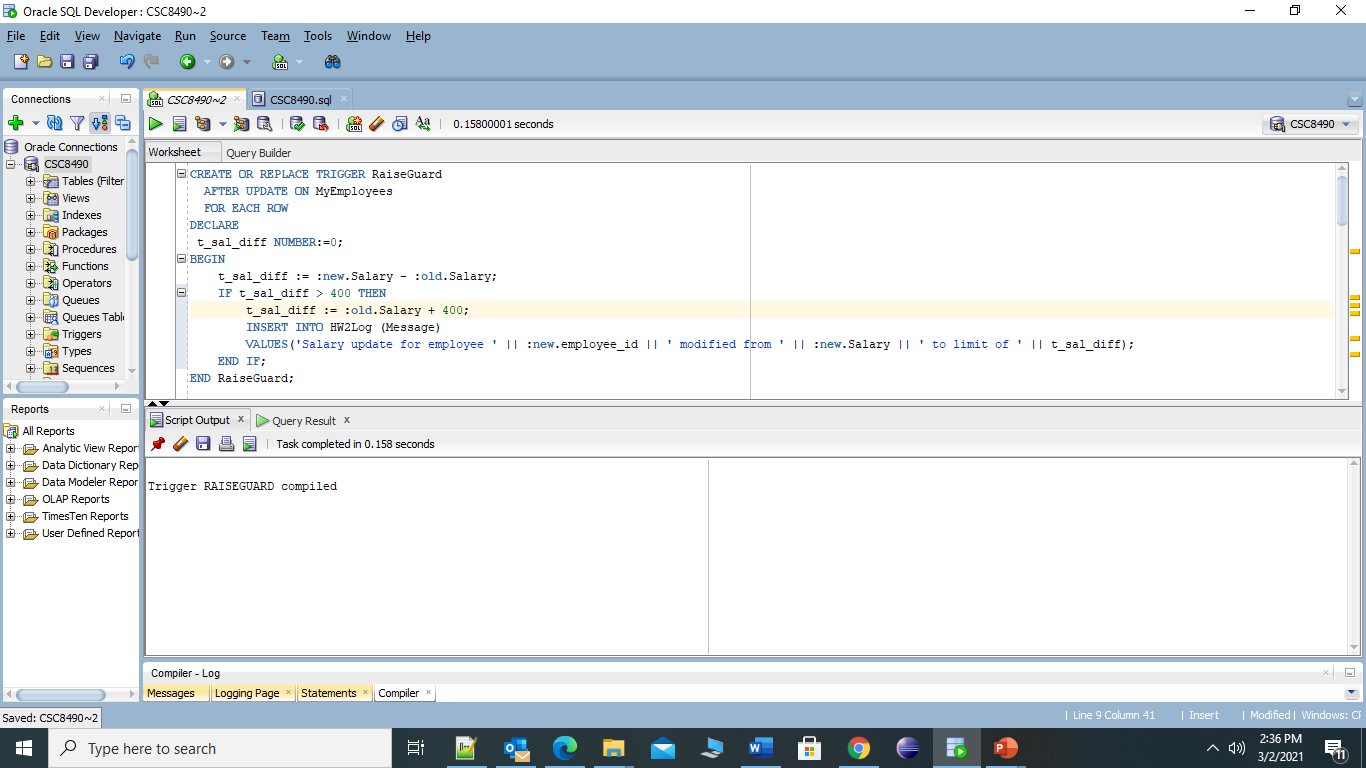
t\_sal\_diff := :old.Salary + 400;

INSERT INTO HW2Log (Message)

VALUES('Salary update for employee ' || :new.employee\_id || ' modified from ' || :new.Salary || ' to limit of ' || t\_sal\_diff);

END IF;

END RaiseGuard;



* Make sure to drop and recreate the MyEmployees table as was described above.

CREATE TABLE HW2Log

(

Message VARCHAR2(200),

TStamp TIMESTAMP DEFAULT SYSTIMESTAMP

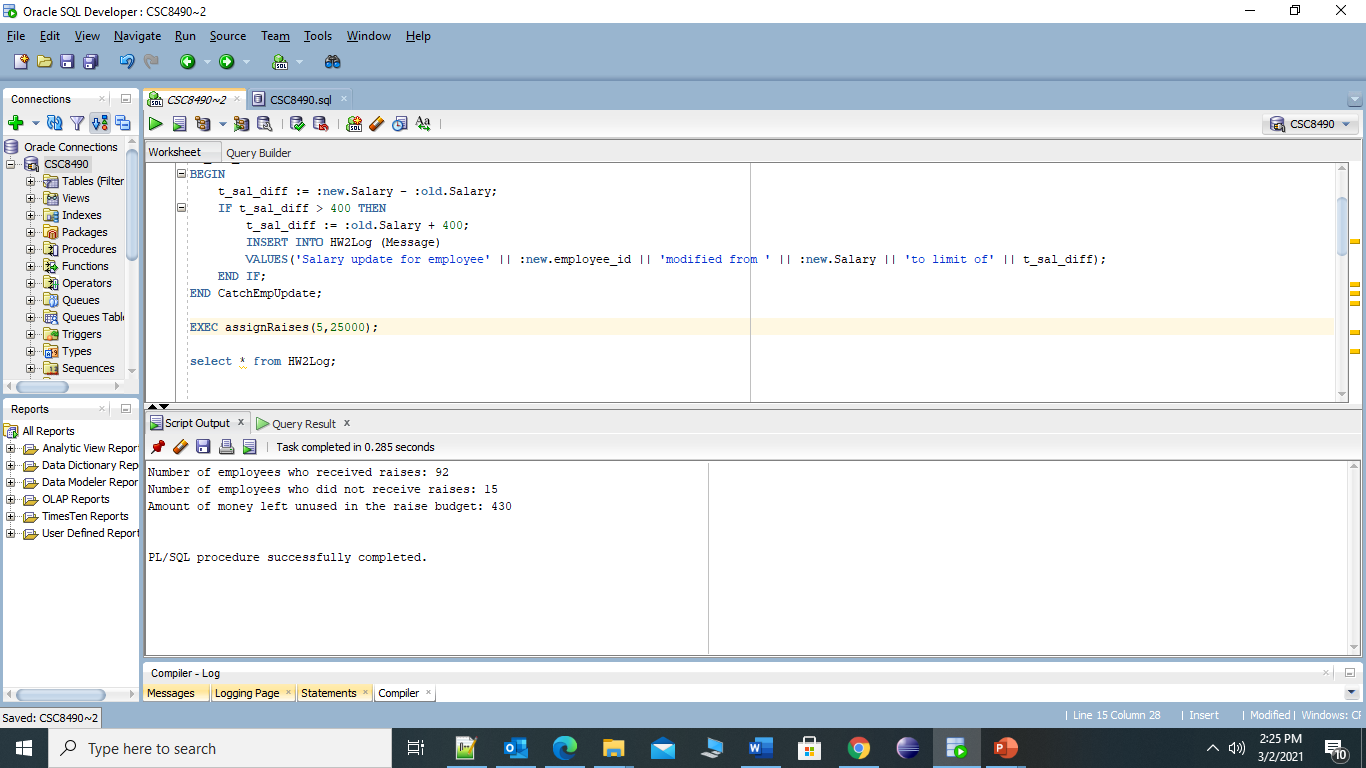
);

DROP TABLE MyEmployees;

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

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

EXEC assignRaises(5,25000);



* Show all of the data from your HW2Log table (can do this with a standard SQL query) – order your results by ascending TStamp value.

**SELECT \* FROM HW2Log ORDER BY TStamp ASC**

**Result Set of above query;**

Salary update for employee 121 modified from 8610 to limit of 8600 02-MAR-21 07.28.31.033377000 PM

Salary update for employee 110 modified from 8610 to limit of 8600 02-MAR-21 07.28.31.034282000 PM

Salary update for employee 206 modified from 8715 to limit of 8700 02-MAR-21 07.28.31.034623000 PM

Salary update for employee 177 modified from 8820 to limit of 8800 02-MAR-21 07.28.31.034958000 PM

Salary update for employee 176 modified from 9030 to limit of 9000 02-MAR-21 07.28.31.035265000 PM

Salary update for employee 175 modified from 9240 to limit of 9200 02-MAR-21 07.28.31.035547000 PM

Salary update for employee 109 modified from 9450 to limit of 9400 02-MAR-21 07.28.31.035864000 PM

Salary update for employee 158 modified from 9450 to limit of 9400 02-MAR-21 07.28.31.036168000 PM

Salary update for employee 152 modified from 9450 to limit of 9400 02-MAR-21 07.28.31.036448000 PM

Salary update for employee 103 modified from 9450 to limit of 9400 02-MAR-21 07.28.31.036719000 PM

Salary update for employee 157 modified from 9975 to limit of 9900 02-MAR-21 07.28.31.037031000 PM

Salary update for employee 151 modified from 9975 to limit of 9900 02-MAR-21 07.28.31.037331000 PM

Salary update for employee 163 modified from 9975 to limit of 9900 02-MAR-21 07.28.31.037596000 PM

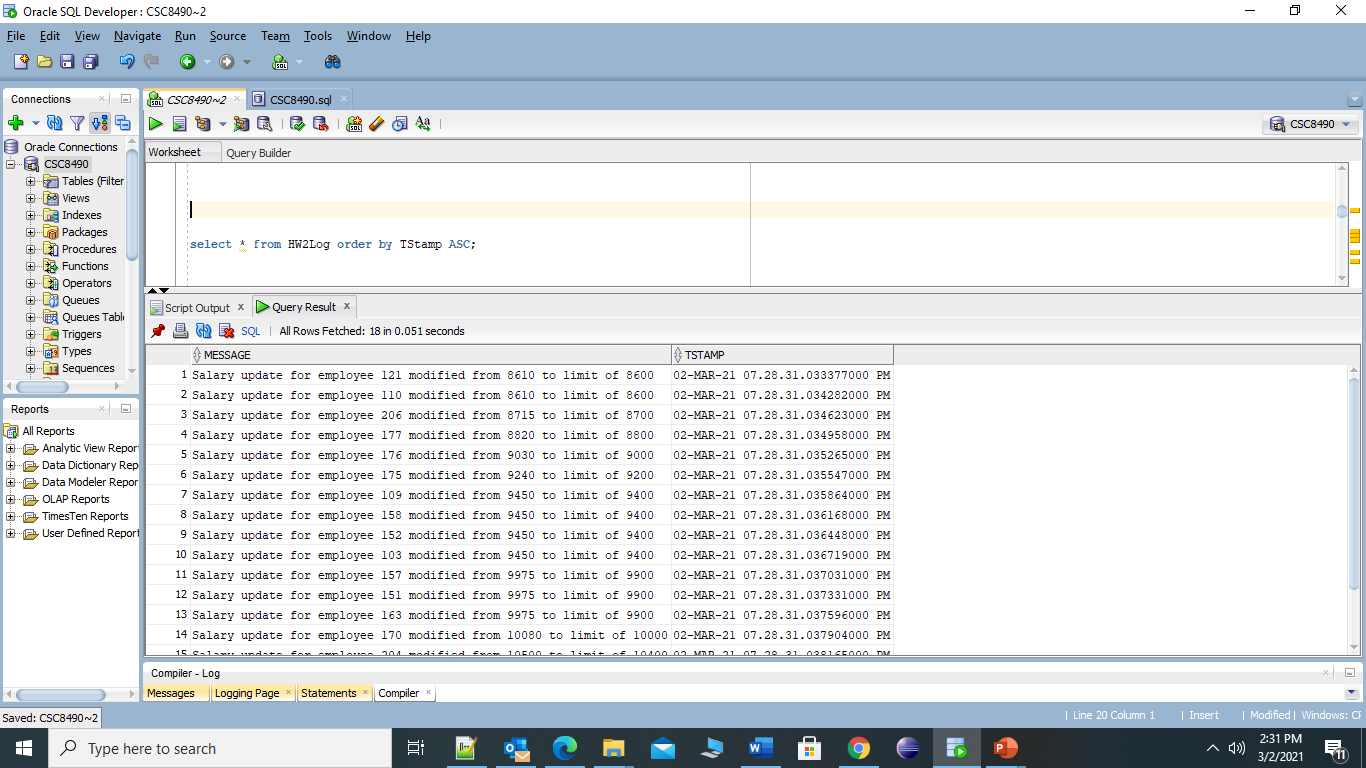
Salary update for employee 170 modified from 10080 to limit of 10000 02-MAR-21 07.28.31.037904000 PM

Salary update for employee 204 modified from 10500 to limit of 10400 02-MAR-21 07.28.31.038165000 PM

Salary update for employee 156 modified from 10500 to limit of 10400 02-MAR-21 07.28.31.038455000 PM

Salary update for employee 150 modified from 10500 to limit of 10400 02-MAR-21 07.28.31.038726000 PM

Salary update for employee 169 modified from 10500 to limit of 10400 02-MAR-21 07.28.31.039115000 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;**

**Result set of above Query :**

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

