Fachbereich 07 Informatik/Mathematik

Praktikum Datenbanksysteme II
Wintersemester 2018/19

Prof. Dr. Martin Staudt

Übung 3

Wimmer, Anja
IF8
Gabl, Daniel
IF6

21.01.2019

Inhaltsverzeichnis

Aufgaben

Aufgabe 1

-

Aufgabe 2

Bei der Implementierung der Min-Max-Skallierung haben wir zuerst eine Tabelle von Personen aufgestellt und mit Test-Werten gefüttert. Als zu skallierenden Wert haben wir das Alter der Person genommen (auch wenn dies nicht wirklich viel Sinn ergibt).

Die Skallierungsfunktion haben wir auch als eigene Funktion definiert, die eben die fünf Parameter (aktueller Wert, altes Minimum, altes Maximum, neues Minimum, neues Maximum) übergeben bekommt und aus der gegebenen Formel das neue Minum berechnet.

Formel d. Min-Max-Skallierung: $v' = \frac{v - old\_min}{old\_max - old\_min}(new\_max - new\_\min) + new\_min$

Beispiel:
Wir wollen eine Skala, die ursprünglich von 17 bis 74 ging, neuskallieren. Die neue Skala soll nun von 10 bis 50 gehen, als aktuellen Wert bekommen wir 34.

$$v' = \frac{34 - 17}{74 - 17}(50 - 10) + 10 = \frac{17}{57}(40) + 10 = \frac{680}{57} + 10 = \frac{1250}{57} = 21.9298 \ldots \approx 22$$

Dazu haben wir eine Prodezur geschrieben, die nur das neue Minimum und das neue Maximum übergeben bekommt, das alte Minimum und das alte Maximum wird mittels den Aggregatfunktionen MIN und MAX aus der Tabelle berechnet. Danach wird auf jeden Datensatz der Tabelle die Min-Max-Skallierung mit den übergebenen und berechneten Daten angewandt, wobei der aktuelle Wert dem Alter im aktuellen Datensatz entspricht. Nach der Berechnung des neuen Werts wird der alte Wert mit dem neuen Wert überschrieben.

Hier Screendumps unserer Tablle vor der Skallierung und nach einer 10,50-Skallierung:

	PERSON_ID	PERSON_NAME	PERSON_AGE	PERSON_PLACE
1	0	Hans	34	Bonn
2	1	Werner	18	Berlin
3	2	Jürgen	74	München
4	3	Luis	25	Augsburg
5	4	Nicklas	22	Regensburg
6	5	Jens	49	Prag
7	6	Marvin	17	Cottbus
8	7	Sophia	22	München
9	8	Günther	63	Ulm
10	9	Sebastian	19	Ingolstadt

	PERSON_ID	PERSON_NAME	PERSON_AGE	PERSON_PLACE
1	0	Hans	22	Bonn
2	1	Werner	11	Berlin
3	2	Jürgen	50	München
4	3	Luis	16	Augsburg
5	4	Nicklas	14	Regensburg
6	5	Jens	32	Prag
7	6	Marvin	10	Cottbus
8	7	Sophia	14	München
9	8	Günther	42	Ulm
10	9	Sebastian	11	Ingolstadt

Als Anmerkung: Eine sinnvolle Anwendung einer Min-Max-Skallierung ist, wenn man bspw. den „Fortschritt“ des aktuellen Jahres in Prozent berechnen möchte (26.12 = 98.36%).

Der Unix-Timestamp vom 01.01, Mitternacht des aktuellen Jahres = altes Minimum,
Der Unix-Timestamp vom 01.01, Mitternacht des nächsten Jahres = altes Maximum,
0 = neue Minimum, 100 = neue Maximum & der aktuelle Unix-Timestamp = aktueller Wert.

$$26.12, 00{:}30\ Uhr = \frac{1545780600 - 1514761200}{1546297200 - 1514761200}(100 - 0) + 0 = 98.361872146 \ldots$$

Aufgabe 3

Aufgabe ist es, unser Data Warehouse mit Daten aus anderen Quellen zu füllen, dabei müssen auch Aktualisierungen korrekt gehandhabt werden.

Zuerst haben wir die vorgegeben Tabellen (Quellen und Zieltabelle) erzeugt und die Quelltabellen mit Beispieldatensätzen gefüttert. Dabei ist uns aufgefallen, dass wir einige Attribute der Zieltabelle nicht in jedem Fall oder nicht in der gewünschten Form haben und wir diese erst noch erzeugen oder uns zurechtlegen müssen, Beispiele dafür:

- Arbeiter haben kein Geschlechter-Attribut
- Angestellte haben ein Attribut, in dem Vor- und Nachname zusammenhängen
- Angestellte und Arbeiter haben nur ein Geburtsdatum, kein Alter
- Angestellte und Arbeiter haben nur Stunden / Monatslohn, kein Jahreseinkommen

Um diese Probleme zu beheben haben wir Hilfstabellen angelegt und Funktionen / Prozeduren geschrieben, die bspw. aus einem Geburtsdatum ein Alter errechnet.

Um das Geschlecht eines Arbeiters bestimmen zu können, haben wie eine Tabelle angelegt, die einen Vornamen auf ein Geschlecht mappt. Wenn wir einen Angestellten in unser Data Warehouse einfügen, nehmen wir seinen Vornamen und sein Geschlecht und speichern es in dieser Hilfstabelle. Fügen wir nun einen Arbeiter hinzu und der Vorname steht in unserer Hilfstabelle, so können wir daraus schließen, welches Geschlecht dieser hat. Wichtig hierbei ist es, dass der Name genau einmal in der Hilfstabelle vorkommt, denn sollte der Name doppelt vorkommen, so können wir wieder keine Aussage treffen.

Um den zusammengesetzten Namen eines Angestellten in Vor- und Nachname zu trennen, haben wir eine Prozedur geschrieben, die mittels einer Substring- und charAt-Funktion den Namen korrekt trennt. Als Format haben wir hierfür „<<Name>>, <<Vorname>>“ angegeben. Es erfordert hierfür eine Prozedur, da eine Funktion nur einen Rückgabewert hat, wir aber zwei Werte brauchen. Um das zu erreichen haben wir die Prozedur mit drei Parametern versehen, dem zusammengesetzten Namen als Input, eine Output-Variable in der der Vorname gespeichert wird und gleiches für den Nachnamen.

Um das Alter einer Person zu berechnen haben wir eine bzw. zwei Funktionen geschrieben, die das Geburtsdatum in der jeweiligen Form (YY/MM/DD für Angestellte, MM.YY für Arbeiter) übergeben bekommt. Zunächst werden (Tag,) Monat und Jahr, wie bei der Prozedur zur Namenstrennung, getrennt und dann wird mit den separierten Werten weitergerechnet.

Hier unser Algorithmus in Pseudocode:

```
IF ((aktuelles Jahr - Geburtsjahr) <= 0)
      Alter = Differenz + 100;  // Wurde im letzten Jahrhundert geboren
ELSE
      Alter = Differenz;  // Dieses Jahrhundert geboren

IF ((aktueller Monat - Geburtsmonat) < 0)
      Alter -= 1;  // Hatte dieses Jahr noch nicht Geburtstag → abziehen
ELSE  IF (Differenz == 0)  //  Hat diesen Monat Geburtstag
                           → Tag berücksichtigen wenn möglich
            IF ((aktueller Tag - Geburtstag) < 0)
                  Alter -= 1;  // Hatte dieses Jahr noch nicht Geburtstag
                           → abziehen

RETURN Alter;
```

Um das korrekte Jahreseinkommen zu berechnen, haben wir erst nach Möglichkeiten gesucht, den Stundenlohn adäquat umzurechnen, da uns $Stundenlohn * 8h * 5d * 4w * 12m$ nicht wirklich absolut korrekt vorgekommen sind. Als Quelle haben wir Arbeitsrechte.de genommen, da die Webseite einen seriößen Eindruck machte. Diese berechnen die Arbeitstage pro Monat unter https://www.arbeitsrechte.de/arbeitstage-pro-monat/ wie folgt:

$$Arbeitsstunden/Mo = Tägliche\ Arbeitszeit * Anzahl\ der\ Arbeitstage/Woche) * 4.3$$

Beispiel für den Normalfall (8h/d, 5d/w): $Arbeitsstunden = 8 * 5 * 4.3 = 172h/Monat$

Um das korrekte Jahreseinkommen aus dem Stundenlohn zu berechnen, nehmen wir o. g. Formel, multiplizieren sie mit 12 Monaten und multiplizieren das mit dem Stundenlohn.
Um das korrekte Jahreseinkommen aus dem Monatsgehalt zu berechnen, nehmen wir einfach diesen und multiplizieren ihn mit 12 Monaten.

Nun haben wir schon fast alle Werte für die Zieltabelle (Nachname, Vorname, Alter, Geschlecht und Jahreseinkommen), alles was noch fehlt sind Personalnummer und Jobcode.

Wir nutzen einen Jobcode (und dazu eine Mapping-Tabelle, die den Code auf eine Berufsbezeichnung mappt), da wir davon ausgehen können, dass einige Berufsbezeichnungen mehrfach vorkommen werden und wir so Speicherplatz sparen können. Wenn wir einen (neuen) Angestellten zu unserem Data Warehouse hinzufügen wollen, prüfen wir, ob es bereits schon einen Code für dessen Berufsbezeichnung gibt, wenn nicht erstellen wir einen 5-stelligen per Zufall generierten Code. Dabei wird auch darauf geachtet, dass dieser Code nicht bereits existiert, sollte das passieren, wird das Prozedere solange wiederholt.

Ähnliches gilt für die Personalnummer. Wenn wir einen neuen Angestellten oder Arbeite hinzufügen wollen prüfen wir, ob dieser bereits in unserem Warehouse ist und wenn nicht, dann generieren wir einen 6-stelligen per Zufall generierten Code. Wie auch schon bei den Jobcodes wird auch hier geprüft, ob diese Personalnummer bereits existiert.

Nun stellte sich uns noch die Frage, wie wir prüfen können, ob es einen Angestellten oder Arbeiter bereits in unserem Data Warehouse gibt. Hierzu haben wir eine weitere Hilfstabelle erstellt, die die Personalnummer, den Namen der Quelltabelle und den alten Primärschlüssel speichert, damit wir den Angestellten / Arbeiter später auch identifizieren können. Beim Angestellten haben wir eine Angestelltennummer, die wir benutzen können, bei den Arbeitern wiederum haben wir das nicht, wir haben den Vor- und Zunamen zur Identifikation genutzt.

Halten wir fest. Wir haben folgende drei Hilfstabellen:

- Eine Tabelle, die einen Vornamen auf einem Geschlecht mappt
- Eine Tabelle, die eine ID auf eine Berufsbezeichnung mappt
- Eine Tabelle, die festhält, ob wir einen Eintrag schon im Data Warehouse haben

Dazu noch folgende Annahmen:

- Name und Vorname kommt in der Kombination bei einem Arbeiter nur einmal vor
- Arbeiter können 1 Jahr und maximal 100 Jahre alt sein
- Ein Arbeiter arbeite 172 Stunden im Monat, es gibt kein Weihnachtsgehalt
- Es werden zuerst Angestellte und dann Arbeiter hinzugefügt

Anbei noch einige Screendumps der Tabellen, um die Korrektheit des Programms zu zeigen:

Vor dem Zusammenführen der Tabellen:
Angestellten-Tabelle:

	EMP_NR	NAME	DOB	JOB_TITLE	SALARY_MONTH	GENDER
1	0	Holmes, Sherlock	81/07/12	Consulting Detective	4500	männlich
2	1	Harkness, Jack	56/03/21	Captain	6600	männlich
3	2	Duck, Dagobert	98/02/14	Millionair	30000	männlich
4	3	Star, Patrick	01/10/31	wasting time	30	männlich
5	4	Everdeen, Katness	93/04/04	leading the Revolution	1500	weiblich
6	5	Weasley, George	78/04/01	Prankster	5000	männlich
7	6	Graham, Will	88/08/08	Empath	2400	männlich
8	7	Kirrin, George	05/11/09	Hobby Detective	5600	weiblich
9	8	Kebekus, Carolin	80/05/09	Comedian	8000	weiblich
10	9	Paige, Ellen	87/02/21	Actor	12000	weiblich

Arbeiter-Tabelle:

	SURNAME	FORENAME	BIRTH_MONTH	SALARY_HOUR
1	Bauer	Hans	04.96	50
2	Fischer	Rainer	10.84	150
3	Paul	George	08.76	160
4	Socher	Gudrun	12.78	135
5	Duck	Donald	07.31	75
6	Jane	Patrick	07.69	95
7	Sparrow	Jack	01.94	200
8	Maier	Carolin	04.90	105
9	Wischhof	Lars	02.75	140
10	Lupo	Chris	08.72	120

Nun wird die Prozedur zur Zusammenführung ausgeführt.
Die Personal-Tabelle sieht wie folgt aus:

	STAFF_NR	SURNAME	FORENAME	AGE	GENDER	JOB_CODE	INCOME_YEAR
1	823182	Holmes	Sherlock	37	2	93937	54000
2	679665	Harkness	Jack	62	2	75339	79200
3	336817	Duck	Dagobert	20	2	41942	360000
4	394888	Star	Patrick	17	2	38118	360
5	323838	Everdeen	Katness	25	1	92828	18000
6	514933	Weasley	George	40	2	98257	60000
7	586949	Graham	Will	30	2	86261	28800
8	618700	Kirrin	George	13	1	34803	67200
9	787287	Kebekus	Carolin	38	1	55825	96000
10	373325	Paige	Ellen	31	1	28080	144000
11	484057	Bauer	Hans	22	0	78536	103200
12	787459	Fischer	Rainer	34	0	78536	309600
13	877990	Paul	George	42	0	78536	330240
14	413393	Socher	Gudrun	40	0	78536	278640
15	384329	Duck	Donald	87	0	78536	154800
16	306618	Jane	Patrick	49	2	78536	196080
17	683128	Sparrow	Jack	25	2	78536	412800
18	313189	Maier	Carolin	28	1	78536	216720
19	945279	Wischhof	Lars	43	0	78536	288960
20	816385	Lupo	Chris	46	0	78536	247680

Man beachte, dass George Paul kein Geschlecht zugewiesen wurde, da „George“ einmal als weiblich und einmal als männlich in der Geschlechter-Tabelle steht.

Die Hilfstabellen haben folgende Einträge:
Geschlechts-Mapping-Tabelle:

	FORENAME	GENDER
1	Sherlock	2
2	Jack	2
3	Dagobert	2
4	Patrick	2
5	Katness	1
6	George	2
7	Will	2
8	George	1
9	Carolin	1
10	Ellen	1

JobCode-Tabelle:

	CODE	DESCRIPTION
1	93937	Consulting Detective
2	75339	Captain
3	41942	Millionair
4	38118	wasting time
5	92828	leading the Revolution
6	98257	Prankster
7	86261	Empath
8	34803	Hobby Detective
9	55825	Comedian
10	28080	Actor
11	78536	worker

Identifizierungs-Tabelle:

	STAFF_NR	SOURCE_TABLE	PK
1	823182	e	0
2	679665	e	1
3	336817	e	2
4	394888	e	3
5	323838	e	4
6	514933	e	5
7	586949	e	6
8	618700	e	7
9	787287	e	8
10	373325	e	9
11	484057	w	Hans Bauer
12	787459	w	Rainer Fischer
13	877990	w	George Paul
14	413393	w	Gudrun Socher
15	384329	w	Donald Duck
16	306618	w	Patrick Jane
17	683128	w	Jack Sparrow
18	313189	w	Carolin Maier
19	945279	w	Lars Wischhof
20	816385	w	Chris Lupo

Nun ändern wir einige Datensätze ab, fügen neue hinzu und führen die Prozedur erneut aus.

Neu:
2 Angestellte (Hannibal Lecter (Angestellter 10) und John Watson (Angestellter 11))
1 Arbeiter (Will Smith, geboren im September 68, hat einen Stundenlohn von 125($))

Verändert:
2 Angestellte:

- Dagobert Duck, ist nun Milliardär und nicht mehr Millionär, außerdem wurde das Geburtsdatum von 14.02.98 auf 14.02.89 geändert (Zahlendreher)
- Sherlock Holmes, verdient nun 5.400($) und nicht mehr 4.500($)

1 Arbeiter:

- Rainer Fischer, dieser hat nicht mehr im Oktober sondern im Mai Geburtstag

Nach dem Update:
Angestellten-Tabelle:

	EMP_NR	NAME	DOB	JOB_TITLE	SALARY_MONTH	GENDER
1	0	Holmes, Sherlock	81/07/12	Consulting Detective	5400	männlich
2	1	Harkness, Jack	56/03/21	Captain	6600	männlich
3	2	Duck, Dagobert	89/02/14	Billionair	30000	männlich
4	3	Star, Patrick	01/10/31	wasting time	30	männlich
5	4	Everdeen, Katness	93/04/04	leading the Revolution	1500	weiblich
6	5	Weasley, George	78/04/01	Prankster	5000	männlich
7	6	Graham, Will	88/08/08	Empath	2400	männlich
8	7	Kirrin, George	05/11/09	Hobby Detective	5600	weiblich
9	8	Kebekus, Carolin	80/05/09	Comedian	8000	weiblich
10	9	Paige, Ellen	87/02/21	Actor	12000	weiblich
11	10	Lecter, Hannibal	75/11/02	Psychiatrist	8000	männlich
12	11	Watson, John	88/07/23	Doctor	4300	männlich

Arbeiter-Tabelle:

	SURNAME	FORENAME	BIRTH_MONTH	SALARY_HOUR
1	Bauer	Hans	04.96	50
2	Fischer	Rainer	05.84	150
3	Paul	George	08.76	160
4	Socher	Gudrun	12.78	135
5	Duck	Donald	07.31	75
6	Jane	Patrick	07.69	95
7	Sparrow	Jack	01.94	200
8	Maier	Carolin	04.90	105
9	Wischhof	Lars	02.75	140
10	Lupo	Chris	08.72	120
11	Smith	Will	09.68	125

Und nun wird die Prozedur erneut ausgeführt.

Die Personal-Tabelle sieht nun wie folgt aus:

	STAFF_NR	SURNAME	FORENAME	AGE	GENDER	JOB_CODE	INCOME_YEAR
1	823182	Holmes	Sherlock	37	2	93937	64800
2	679665	Harkness	Jack	62	2	75339	79200
3	336817	Duck	Dagobert	29	2	61850	360000
4	394888	Star	Patrick	17	2	38118	360
5	323838	Everdeen	Katness	25	1	92828	18000
6	514933	Weasley	George	40	2	98257	60000
7	586949	Graham	Will	30	2	86261	28800
8	618700	Kirrin	George	13	1	34803	67200
9	787287	Kebekus	Carolin	38	1	55825	96000
10	373325	Paige	Ellen	31	1	28080	144000
11	484057	Bauer	Hans	22	0	78536	103200
12	787459	Fischer	Rainer	34	0	78536	309600
13	877990	Paul	George	42	0	78536	330240
14	413393	Socher	Gudrun	40	0	78536	278640
15	384329	Duck	Donald	87	0	78536	154800
16	306618	Jane	Patrick	49	2	78536	196080
17	683128	Sparrow	Jack	25	2	78536	412800
18	313189	Maier	Carolin	28	1	78536	216720
19	945279	Wischhof	Lars	43	0	78536	288960
20	816385	Lupo	Chris	46	0	78536	247680
21	675711	Lecter	Hannibal	43	2	18726	96000
22	382330	Watson	John	30	2	59811	51600
23	389430	Smith	Will	50	2	78536	258000

Man beachte, dass

- das Jahreseinkommen von Sherlock Holmes nun 64.800 und nicht mehr 54.000 ist.
- Dagobert Duck einen anderen Jobcode hat und er nicht mehr 20 sondern 29 ist.
- Will Smith als männlich eingetragen ist, da „Will“ in der Gender-Tabelle ist.
- das Alter von Rainer Fischer auch weiterhin bei 34 ist, da gerade Januar ist.
- die bereits existierenden Einträge nicht doppelt in unserer Tabelle sind.

Änderungen in den Hilfstabellen:

Geschlechter:

	FORENAME	GENDER
1	Sherlock	2
2	Jack	2
3	Dagobert	2
8	George	1
9	Carolin	1
10	Ellen	1
11	Hannibal	2
12	John	2

Job-Codes:

	CODE	DESCRIPTION
1	93937	Consulting Detective
2	75339	Captain
3	41942	Millionair
4	38118	wasting time
5	92828	leading the Revolution
11	78536	worker
12	61850	Billionair
13	18726	Psychiatrist

Identifizierung:

	STAFF_NR	SOURCE_TABLE	PK
1	823182	e	0
2	679665	e	1
3	336817	e	2
4	394888	e	3
5	323838	e	4
20	816385	w	Chris Lupo
21	675711	e	10
22	382330	e	11
23	389430	w	Will Smith

Quellcode

Die SQL-Files und die Doku finden Sie zusätzlich auf GitHub.com/AnyaW/DBS2.

Min-Max-Skalierung:

```
CREATE TABLE People (
      person_id INT NOT NULL,
      person_name VARCHAR(63),
      person_age INT,
      person_place VARCHAR(63),
      CONSTRAINT pers_id PRIMARY KEY (person_id)
);

INSERT INTO People VALUES(0, 'Hans', 34, 'Bonn');
INSERT INTO People VALUES(1, 'Werner', 18, 'Berlin');
INSERT INTO People VALUES(2, 'Jürgen', 74, 'München');
INSERT INTO People VALUES(3, 'Luis', 25, 'Augsburg');
INSERT INTO People VALUES(4, 'Nicklas', 22, 'Regensburg');
INSERT INTO People VALUES(5, 'Jens', 49, 'Prag');
INSERT INTO People VALUES(6, 'Marvin', 17, 'Cottbus');
INSERT INTO People VALUES(7, 'Sophia', 22, 'München');
INSERT INTO People VALUES(8, 'Günther', 63, 'Ulm');
INSERT INTO People VALUES(9, 'Sebastian', 19, 'Ingolstadt');

CREATE OR REPLACE FUNCTION min_max_scale(v IN INT, new_min IN INT,
new_max IN INT, old_min IN INT, old_max IN INT)
      RETURN INT
      IS new_v INT;

BEGIN
      new_v := ((v - old_min) / (old_max - old_min)) * (new_max -
new_min) + new_min; -- min max impl
      RETURN(new_v);

END min_max_scale;
/

CREATE OR REPLACE PROCEDURE MinMax_Scaling (new_min INT, new_max
INT) IS
      old_min INT;
      old_max INT;
      new_age INT;

BEGIN
      SELECT MIN(person_age) as min_age, MAX(person_age) as max_age
INTO old_min, old_max FROM People; -- get min and max

      FOR person IN (SELECT * FROM People) LOOP
            new_age   :=   min_max_scale(person.person_age,   new_min,
new_max, old_min, old_max);
            UPDATE People SET person_age = new_age WHERE person_id =
person.person_id;
      END LOOP;
END;
/

EXECUTE MinMax_Scaling(10, 50);
```

Data Warehouse:
Tables:

```
CREATE TABLE Employee (
     emp_nr VARCHAR(9) NOT NULL,
     name VARCHAR(31), -- surname, forname
     dob VARCHAR(8),  -- yy/mm/dd
     job_title VARCHAR(63),
     salary_month DOUBLE PRECISION,
     gender VARCHAR(9),  -- männlich / weiblich
     CONSTRAINT employee_nr PRIMARY KEY (emp_nr)
);

CREATE TABLE Worker (
     surname VARCHAR(31),
     forename VARCHAR(31),
     birth_month VARCHAR(5),  -- mm.yy
     salary_hour DOUBLE PRECISION
);

CREATE TABLE Staff (
     staff_nr INTEGER NOT NULL,
     surname VARCHAR(31),
     forename VARCHAR(31),
     age INT,
     gender INT,  -- 0 = unknown, 1 = female, 2 = male
     job_code VARCHAR(7),
     income_year DOUBLE PRECISION,
     CONSTRAINT s_nr PRIMARY KEY (staff_nr)
);

CREATE TABLE Jobcode (
     code INTEGER PRIMARY KEY,
     description VARCHAR(63)
);

CREATE TABLE Gender (
     forename VARCHAR(31),
     gender VARCHAR(1),  -- 1 = female, 2 = male
     CONSTRAINT gender_entry PRIMARY KEY (forename, gender)
);

CREATE TABLE Identifier (
     staff_nr INTEGER PRIMARY KEY,
     source_table VARCHAR(1), -- e = Employee, w = Worker
     pk VARCHAR(63) -- Employee: emp_nr, Worker: Forename +
Surname
);
```

Inserts:

```
INSERT INTO Employee VALUES ('0', 'Holmes, Sherlock',
'81/07/12', 'Consulting Detective', 4500.00, 'männlich');
INSERT INTO Employee VALUES ('1', 'Harkness, Jack',
'56/03/21', 'Captain', 6600.00, 'männlich');
INSERT INTO Employee VALUES ('2', 'Duck, Dagobert',
'98/02/14', 'Millionair', 30000.00, 'männlich');
INSERT INTO Employee VALUES ('3', 'Star, Patrick', '01/10/31',
'wasting time', 30.00, 'männlich');
INSERT INTO Employee VALUES ('4', 'Everdeen, Katness',
'93/04/04', 'leading the Revolution', 1500.00, 'weiblich');
INSERT INTO Employee VALUES ('5', 'Weasley, George',
'78/04/01', 'Prankster', 5000.00, 'männlich');
INSERT INTO Employee VALUES ('6', 'Graham, Will', '88/08/08',
'Empath', 2400.00, 'männlich');
INSERT INTO Employee VALUES ('7', 'Kirrin, George',
'05/11/09', 'Hobby Detective', 5600.00, 'weiblich');
INSERT INTO Employee VALUES ('8', 'Kebekus, Carolin',
'80/05/09', 'Comedian', 8000.00, 'weiblich');
INSERT INTO Employee VALUES ('9', 'Paige, Ellen', '87/02/21',
'Actor', 12000.00, 'weiblich');

INSERT INTO Worker VALUES ('Bauer', 'Hans', '04.96', 50.0);
INSERT INTO Worker VALUES ('Fischer', 'Rainer', '10.84',
150.0);
INSERT INTO Worker VALUES ('Paul', 'George', '08.76', 160.0);
INSERT INTO Worker VALUES ('Socher', 'Gudrun', '12.78',
135.0);
INSERT INTO Worker VALUES ('Duck', 'Donald', '07.31', 75.0);
INSERT INTO Worker VALUES ('Jane', 'Patrick', '07.69', 95.0);
INSERT INTO Worker VALUES ('Sparrow', 'Jack', '01.94', 200.0);
INSERT INTO Worker VALUES ('Maier', 'Carolin', '04.90',
105.0);
INSERT INTO Worker VALUES ('Wischhof', 'Lars', '02.75',
140.0);
INSERT INTO Worker VALUES ('Lupo', 'Chris', '08.72', 120.0);
```

Merge:

```
SET SERVEROUTPUT ON;

-- private function
CREATE OR REPLACE FUNCTION generate_id(digits IN INTEGER)
RETURN INTEGER
IS
      new_id INTEGER;
      max_nr INTEGER := 9;
      id_str VARCHAR(10);
      cur_nr INTEGER;
      starts_with_zero BOOLEAN;
BEGIN
      FOR i in 1..digits LOOP
            cur_nr := dbms_random.value(0, max_nr);
            id_str := CONCAT(id_str, cur_nr);
      END LOOP;

      starts_with_zero := INSTR(id_str, '0') = 1;
      IF starts_with_zero THEN
            id_str := CONCAT(id_str, '9');
      END IF;

      new_id := TO_NUMBER(id_str);
      RETURN new_id;
END;
/

CREATE OR REPLACE PROCEDURE get_staffnr(identify IN VARCHAR, source
IN VARCHAR, staffnr OUT INTEGER, exist OUT BOOLEAN)
IS
      results INTEGER;
BEGIN
      SELECT COUNT(*) INTO results FROM Identifier WHERE pk =
identify AND source_table = source;

      IF results = 0 THEN
            staffnr := -1;
            WHILE staffnr = -1 OR results != 0 LOOP
                  staffnr := generate_id(6);
                  SELECT COUNT(*) INTO results FROM Identifier WHERE
staff_nr = staffnr;
            END LOOP;

            exist := FALSE;
            INSERT INTO Identifier VALUES (staffnr, source,
identify);
      ELSE
            SELECT src.staff_nr INTO staffnr FROM Identifier src
WHERE pk = identify;
            exist := TRUE;
      END IF;
END;
/
```

```
CREATE OR REPLACE PROCEDURE split_name(staff_name IN VARCHAR,
surname IN OUT VARCHAR, forename IN OUT VARCHAR)
IS
BEGIN
     surname  := '';
     forename := '';
     SELECT SUBSTR(staff_name, 1, INSTR(staff_name, ',', 1, 1) - 1),
             SUBSTR(staff_name, INSTR(staff_name, ',', 1, 1) + 2)
     INTO surname, forename
     FROM DUAL;
END;
/

CREATE OR REPLACE FUNCTION concat_name(forename IN VARCHAR, surname
IN VARCHAR)
RETURN VARCHAR
IS
     fullname VARCHAR(64);
BEGIN
     fullname := CONCAT(CONCAT(forename, ' '), surname);
     RETURN fullname;
END;
/

CREATE OR REPLACE PROCEDURE split_date_emp(dt IN VARCHAR, yy IN OUT
INTEGER, mm IN OUT INTEGER, dd IN OUT INTEGER)
IS
BEGIN
     yy := 0;
     mm := 0;
     dd := 0;

     -- SUBSTR takes a String, the position of the Beginning and
optional the length
     -- INSTR takes a haystack, a needed, the starting index and the
occurrence

     -- String from position 1 (first char) going to first
occurrence of '/' - 1 = year
     SELECT TO_NUMBER(SUBSTR(dt, 1, INSTR(dt, '/', 1, 1) - 1)),
          -- String from the first occurrence of '/' + 1 going for
the amount of chars
          -- between the second and the first occurrence - 1 to
remove the slash = month
          TO_NUMBER(SUBSTR(dt, INSTR(dt, '/', 1, 1) + 1, INSTR(dt,
'/', 1, 2) - INSTR(dt, '/', 1, 1) - 1)),
          -- String from the second occurrence of '/' going to the
end of the String = day
          TO_NUMBER(SUBSTR(dt, INSTR(dt, '/', 1, 2) + 1))
     INTO yy, mm, dd
     FROM DUAL;
END;
/
```

```
CREATE OR REPLACE FUNCTION dob_age_emp(date_dob IN VARCHAR)
RETURN INTEGER
IS
      age INTEGER;

      year_dob INTEGER;
      month_dob INTEGER;
      day_dob INTEGER;

      year_cur INTEGER;
      month_cur INTEGER;
      day_cur INTEGER;

      date_cur VARCHAR(8);

      year_diff INTEGER;
      month_diff INTEGER;
      day_diff INTEGER;
BEGIN
      -- Get Current Time
      SELECT to_char(sysdate, 'yy/mm/dd') INTO date_cur FROM DUAL;

      -- Split Date into day, month and year
      split_date_emp(date_dob, year_dob, month_dob, day_dob);
      split_date_emp(date_cur, year_cur, month_cur, day_cur);

      age := 0;

      -- diff not positive? Add 100, else it's the difference.
      year_diff := year_cur - year_dob;
      IF (year_diff <= 0) THEN
            age := year_diff + 100;
      ELSE
            age := year_diff;
      END IF;

      -- diff negative? Didn't have his birthday this year so remove
it.
      month_diff := month_cur - month_dob;
      IF (month_diff < 0) THEN
            age := age - 1;
      ELSE
            -- same month? We need to check the day then.
            IF (month_diff = 0) THEN
                  day_diff := day_cur - day_dob;
                  -- diff negative? Didn't have his birthday this year
so remove it.
                  IF (day_diff < 0) THEN
                        age := age - 1;
                  END IF;
            END IF;
      END IF;
      RETURN age;
END;
/
```

```
CREATE OR REPLACE PROCEDURE split_date_wor(dt IN VARCHAR, yy IN OUT
INTEGER, mm IN OUT INTEGER) IS
BEGIN
     mm := 0;
     yy := 0;

     -- SUBSTR takes a String, the position of the Beginning and
optional the length
     -- INSTR takes a haystack, a needed, the starting index and the
occurrence

     --  String from position 1 (first char) going to first
occurrence of '.' - 1 = month
     SELECT TO_NUMBER(SUBSTR(dt, 1, INSTR(dt, '.', 1, 1) - 1)),
          -- String from the first occurrence of '.' + 1 going
until the end = year
          TO_NUMBER(SUBSTR(dt, INSTR(dt, '.', 1, 1) + 1))
     INTO mm, yy
     FROM DUAL;
END;
/

CREATE OR REPLACE FUNCTION dob_age_wor(date_dob IN VARCHAR)
RETURN INTEGER
IS
     year_dob INTEGER;
     month_dob INTEGER;
     year_cur INTEGER;
     month_cur INTEGER;
     date_cur VARCHAR(8);
     year_diff INTEGER;
     month_diff INTEGER;
     age INTEGER;
BEGIN
     -- Get Current Time
     SELECT to_char(sysdate, 'mm.yy') INTO date_cur FROM DUAL;

     --  Split Date into day, month and year
     split_date_wor(date_dob, year_dob, month_dob);
     split_date_wor(date_cur, year_cur, month_cur);

     age := 0;

     -- diff not positive? Add 100, else it's correct already.
     year_diff := year_cur - year_dob;
     IF (year_diff <= 0) THEN
          age := year_diff + 100;
     END IF;

     -- diff negative? Didn't have his birthday yet so remove it.
     month_diff := month_cur - month_dob;
     IF (month_diff < 0) THEN
          age := age - 1;
     END IF;
     RETURN age;
END;
/
```

```
-- this is only for worker
CREATE OR REPLACE FUNCTION gender_mapping(fname IN VARCHAR)
RETURN VARCHAR IS
      answer INTEGER;
      gen VARCHAR(1);
BEGIN
      -- if name only once in TABLE then use this gender -> 1, 2
      -- else gender is unknown -> 0

      SELECT COUNT(*)
      INTO answer
      FROM Gender g
      WHERE g.forename = fname;

      IF answer = 1 THEN
            SELECT g.gender INTO gen
            FROM Gender g
            WHERE g.forename = fname;
      ELSE
            gen := '0';
      END IF;
      RETURN gen;
END;
/

-- this is only for employees
CREATE OR REPLACE FUNCTION gender_evaluation(fname IN VARCHAR, gen
IN VARCHAR)
RETURN VARCHAR IS
      gen_nr VARCHAR(1);
      answer INTEGER;
BEGIN
      -- if e then map m, w to 2, 1
      -- check if already in gender table
      -- if false then insert into TABLE Gender
      -- return 2 or 1
      IF gen = 'weiblich' THEN
            gen_nr := '1';
      ELSE
            IF gen = 'männlich' THEN
                  gen_nr := '2';
            END IF;
      END IF;

      SELECT COUNT(*)
      INTO answer
      FROM Gender g
      WHERE g.forename = fname AND g.gender = gen_nr;

      IF answer = 0 THEN
            INSERT INTO Gender VALUES (fname, gen_nr);
            COMMIT;
      END IF;

      RETURN gen_nr;
END;
/
```

```
CREATE OR REPLACE FUNCTION get_jobcode(job_title IN VARCHAR)
RETURN INTEGER
IS
      job_code INTEGER;
      results INTEGER;
BEGIN
      SELECT COUNT(*) INTO results FROM Jobcode WHERE description =
job_title;

      IF results = 0 THEN
            job_code := -1;
            WHILE job_code = -1 OR results != 0 LOOP
                  job_code := generate_id(5);
                  SELECT COUNT(*) INTO results FROM Jobcode WHERE code
= job_code;
            END LOOP;
            INSERT INTO Jobcode VALUES (job_code, job_title);
      ELSE
            SELECT code INTO job_code FROM Jobcode WHERE description
= job_title;
      END IF;
      RETURN job_code;
END;
/

CREATE OR REPLACE FUNCTION annual_income(income IN DOUBLE PRECISION,
source IN VARCHAR)
RETURN DOUBLE PRECISION
IS
      year_income DOUBLE PRECISION;
BEGIN
      IF (source = 'e') THEN
            year_income := income * 12;
      ELSE
            IF (source = 'w') THEN
                  year_income := income * 12 * (8 * 5 * 4.3);
            END IF;
      END IF;
      RETURN year_income;
END;
/


CREATE OR REPLACE PROCEDURE merge_employees
IS
      -- var
      entry_exist BOOLEAN;

      -- to save
      staffnr INTEGER;
      sname VARCHAR(31);
      fname VARCHAR(31);
      a INTEGER;
      jobcode INTEGER;
      income INTEGER;
      gen VARCHAR(1);
BEGIN
```

```
	FOR employee IN (SELECT * FROM Employee) LOOP
		-- procedures (multiple output values)
		get_staffnr(employee.emp_nr, 'e', staffnr, entry_exist);
		split_name(employee.name, sname, fname); -- split name

		-- functions
		a := dob_age_emp(employee.dob); -- calc age
		gen := gender_evaluation(fname, employee.gender);
		jobcode := get_jobcode(employee.job_title);
		income := annual_income(employee.salary_month, 'e');

		IF entry_exist THEN
			UPDATE Staff s
			SET s.surname = sname, s.forename = fname, s.age =
a, s.gender = gen, s.job_code = jobcode, s.income_year = income
			WHERE s.staff_nr = staffnr;
		ELSE
			INSERT INTO Staff
			VALUES(staffnr, sname, fname, a, gen, jobcode,
income);
		END IF;
	END LOOP;
END;
/

CREATE OR REPLACE PROCEDURE merge_worker
IS
	identify VARCHAR(64);
	exist BOOLEAN;

	staffnr INTEGER;
	a INTEGER;
	gen VARCHAR(1);
	jobcode INTEGER;
	income DOUBLE PRECISION;
BEGIN
	FOR worker IN (SELECT * FROM Worker) LOOP
		identify := concat_name(worker.forename, worker.surname);
		get_staffnr(identify, 'w', staffnr, exist);
		a := dob_age_wor(worker.birth_month);
		gen := gender_mapping(worker.forename);
		jobcode := get_jobcode('worker');
		income := annual_income(worker.salary_hour, 'w');

		IF exist THEN
			UPDATE Staff s
			SET s.age = a, s.gender = gen, s.job_code = jobcode,
s.income_year = income
			WHERE s.staff_nr = staffnr;
		ELSE
			INSERT INTO Staff
			VALUES (staffnr, worker.surname, worker.forename, a,
gen, jobcode, income);
		END IF;
	END LOOP;
END;
/
```

```
CREATE OR REPLACE PROCEDURE merge_staff
IS
BEGIN
     merge_employees();
     merge_worker();
END;
/

EXECUTE merge_staff;
```

Updates:

```
-- Update Tables after first execute

-- Employees
INSERT INTO Employee VALUES ('10', 'Lecter, Hannibal', '75/11/02',
'Psychiatrist', 8000.00, 'männlich');
INSERT INTO Employee VALUES ('11', 'Watson, John', '88/07/23',
'Doctor', 4300.00, 'männlich');

-- dob and job title
UPDATE Employee e
SET e.name = 'Duck, Dagobert', e.dob = '89/02/14',
     e.job_title = 'Billionair',
     e.salary_month = 30000.00, e.gender = 'männlich'
WHERE e.emp_nr = '2';

-- salery
UPDATE Employee e
SET e.name = 'Holmes, Sherlock', e.dob = '81/07/12',
     e.job_title = 'Consulting Detective',
     e.salary_month = 5400.00, e.gender = 'männlich'
WHERE e.emp_nr = '0';


-- Worker
INSERT INTO Worker VALUES ('Smith', 'Will', '09.68', 125.0);
-- dob
UPDATE Worker w
SET w.birth_month = '05.84', w.salary_hour = 150.0
WHERE w.surname = 'Fischer' AND w.forename = 'Rainer';
```