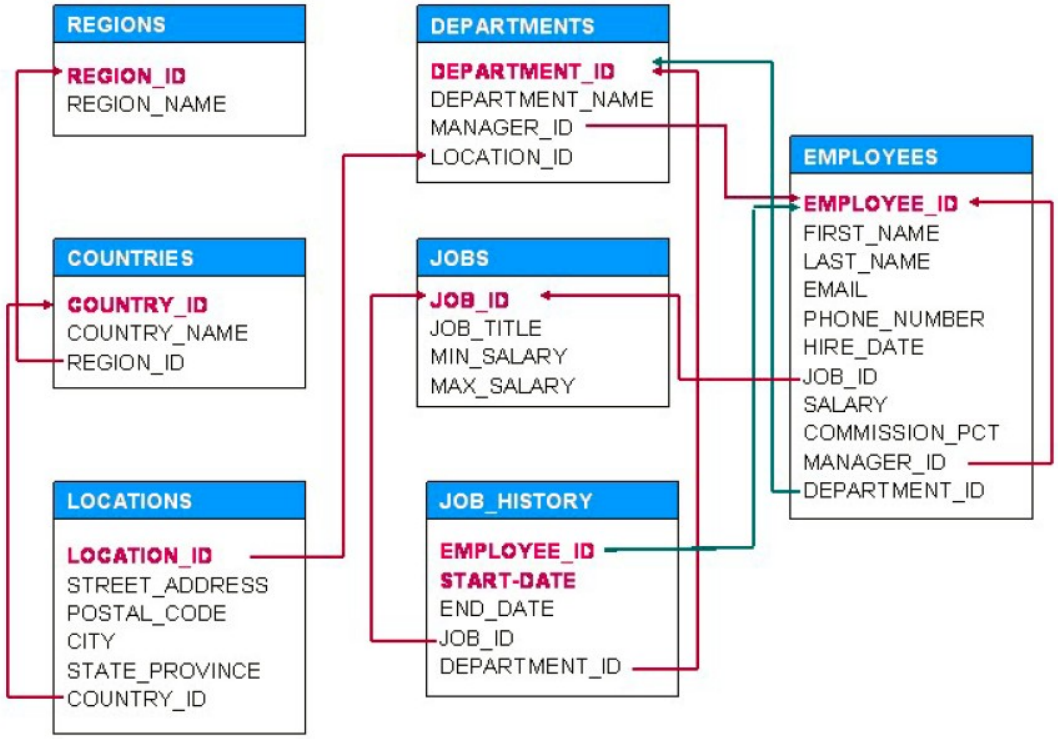
Inhaltsverzeichnis

Abgabe vom Reinsperger (5CHIF) aus der telegram gruppe, einfach doppelt auf das Symbol klicken, dann geht die Word datei auf. (Passt zu keiner testangabe, nur teilweise....)



Übungszettel 1 – Einführung HR

1. Schreiben Sie ein Programm, welches das Einkommen der Mitarbeiter 119 und 124 vertauscht (1L)

```
create or replace PROCEDURE SWAP_SALARY2 (
  p_emp_1 IN employees.employee_id%type,
  p_emp_2 IN employees.employee_id%type
) AS
  v_sal_1 employees.salary%type;
  v_sal_2 employees.salary%type;

BEGIN
  update employees set salary = get_salary_by_id(p_emp_2)
  where employee_id = p_emp_1;

  update employees set salary = get_salary_by_id(p_emp_1)
  where employee_id = p_emp_2;

END SWAP_SALARY2;
```

Dazugehörige Funktion

```
create or replace FUNCTION GET_SALARY_BY_ID
(
  EMP IN employees.employee_id%type
) RETURN employees.salary%type AS
  v_sal employees.salary%type;
BEGIN

  select salary
  into  v_sal
```

```
  from employees
  where employee_id = emp;

  RETURN v_sal;
END GET_SALARY_BY_ID;
```

2. Erhöhen Sie das Einkommen der Mitarbeiter 110 wie folgt:
   - die Firmenzugehörigkeit ist mehr als 7 Jahre -> 25%
   - die Firmenzugehörigkeit ist mehr als 4 Jahre -> 12%
   - in jedem anderen Fall: -> 9%

3. Der Provisionsprozentsatz für den Mitarbeiter 148 soll berechnet werden. Das geht so:
   - wenn das Gehalt mehr als 9000 ist -> 0.45%
   - wenn das Gehalt < 9000, aber die Firmenzugehörigkeit > 7 Jahre ist -> 0.32%
   - wenn das Gehalt <= 2500 -> 0.24%
   - in jedem anderen Fall -> 0.09%.

4. ermitteln Sie den Vorgesetzten des Mitarbeiters 103. Geben Sie den Namen und die Abteilung aus, wo dieser Vorgesetzte arbeitet.

5. Suche Sie nach Lücken und den Mitarbeiter IDs (1L)

```
create or replace PROCEDURE CHECK_IDS AS
  v_uid1 employees.Employee_ID%Type;
  v_uid2 employees.Employee_ID%Type;
  v_uidmax employees.Employee_ID%Type;
  v_uidmin employees.Employee_ID%Type;
BEGIN
  select max(Employee_ID), min(Employee_ID)
  into v_uidmax, v_uidmin
  from employees;
  while(v_uidmin < v_uidmax)
  loop
  select Employee_ID
  into v_uid1
  from employees
  where Employee_ID = v_uidmin;
  select Employee_ID
  into v_uid2
  from employees
  where Employee_ID = v_uidmin + 1;
  if(v_uid2 - v_uid1 != 1)
  then dbms_output.put_line('Eine Lücke zwischen' || v_uid1 || ' und
' || v_uid2  );
  end if;
```

```
  v_uidmin := v_uidmin +1;
  end loop;


  NULL;
END CHECK_IDS;
```

6. Ermitteln Sie das Jahr, in dem die meisten Mitarbeiter eingestellt worden sind. Zeigen Sie - je Monat- die Anzahl der aufgenommenen Mitarbeiter an.

Übungszettel 2 – Cursor HR

1. Zeigen Sie den Job Title und den Namen des Mitarbeiters an, der am längsten in der Firma arbeitet (1L)

```
CREATE OR REPLACE PROCEDURE UE2_A1 AS


 v_FIRST_NAME EMPLOYEES.FIRST_NAME%type;
 v_LAST_NAME EMPLOYEES.LAST_NAME%type;
 v_JOB_TITLE JOBS.JOB_TITLE%type;

BEGIN
  select FIRST_NAME, LAST_NAME, JOB_TITLE
  into v_FIRST_NAME, v_LAST_NAME, v_JOB_TITLE
  from (select FIRST_NAME, LAST_NAME, JOB_TITLE
    from EMPLOYEES
    inner join JOBS on ( EMPLOYEES.JOB_ID = JOBS.JOB_ID)
    order by hire_date)
  where ROWNUM = 1;

    DBMS_OUTPUT.put_line(v_FIRST_NAME || ' ' || v_LAST_NAME || ', '
|| v_JOB_TITLE);
END UE2_A1;

-- Steven King, President
```

2. Zeigen Sie den 2. bis zum 9 Mitarbeiter in der Mitarbeiter Tabelle an. (3L)

```
CREATE OR REPLACE PROCEDURE UE2_A2 AS

  cursor employee_cur is
    select FIRST_NAME, LAST_NAME
    from employees
    where rownum between 2 and 9;

  v_employee employee_cur%ROWTYPE;

BEGIN
  OPEN employee_cur;

  LOOP
    FETCH employee_cur INTO v_employee;
    EXIT WHEN employee_cur%NOTFOUND;

    DBMS_OUTPUT.put_line(v_employee.FIRST_NAME || ' ' ||
v_employee.LAST_NAME );

  END LOOP;
  CLOSE employee_cur;
END UE2_A2;
```

```
create or replace PROCEDURE SHOW_EMP_2_9 AS
  CURSOR c_emp is
    select employee_id
```

```
    from employees
    order by 1;

BEGIN
  FOR v_emp_record IN c_emp
    LOOP
      EXIT WHEN c_emp%ROWCOUNT > 9;
      if (c_emp%ROWCOUNT > 1) then
         DBMS_OUTPUT.PUT(c_emp%ROWCOUNT);
         DBMS_OUTPUT.PUT(', ');
         DBMS_OUTPUT.PUT_LINE(v_emp_record.employee_id);
      end if;
    END LOOP;
END SHOW_EMP_2_9;
```

```
create or replace PROCEDURE ZEIGEMITARBEITER AS

  v_id employees.employee_id%type;
  cursor cur is
  select employees.employee_id from employees
  where rownum > 2;
BEGIN

  open cur;
    loop
      fetch cur
      into v_id;
      exit when cur%RowCount > 9;
      dbms_output.put_line(v_id);
    end loop;
  close cur;

  NULL;
END ZEIGEMITARBEITER;
```

3. Geben Sie den Mitarbeitern eine Gehaltserhöhung: (1L)

- wenn die Abteilung 40 ist -> 9 %
- wenn die Abteilung 70 ist -> 17%
- wenn der Provisionsprozentsatz (commission_prct) > 0,35% ist -> 4%
- in jedem anderen Fall ->11%

```
CREATE OR REPLACE PROCEDURE UE2_A3 AS

  cursor employee_cur is
    select *
    from employees;

  v_employee employee_cur%ROWTYPE;

BEGIN
  OPEN employee_cur;

  LOOP
    FETCH employee_cur INTO v_employee;
    EXIT WHEN employee_cur%NOTFOUND;

    if v_employee.DEPARTMENT_ID = 40 then
      SET_SALARY( v_employee.employee_id, v_employee.salary * 1.09);
    elsif v_employee.DEPARTMENT_ID = 70 then
      SET_SALARY( v_employee.employee_id, v_employee.salary * 1.17);
    elsif v_employee.COMMISSION_PCT > 0.35 then
      SET_SALARY( v_employee.employee_id, v_employee.salary * 1.04);
    else
      SET_SALARY( v_employee.employee_id, v_employee.salary * 1.11);
    end if;

  END LOOP;
  CLOSE employee_cur;
  NULL;
END UE2_A3;
```

Übungszettel 3 – Functions HR

1. Erstellen Sie eine Funktion, die als Parameter die Abteilungsnummer hat und den Namen des Managers liefert. (1L)

```
CREATE OR REPLACE FUNCTION "5BHIF20167_04"."FUNCTIONS03_1_GREILINGER"
(
    v_dep_id IN INT
)RETURN VARCHAR2 AS
  v_fn VARCHAR2(20);
  v_ln VARCHAR2(20);
BEGIN
  SELECT EMPLOYEES.LAST_NAME, EMPLOYEES.FIRST_NAME
    INTO v_ln, v_fn
  FROM EMPLOYEES INNER JOIN DEPARTMENTS ON EMPLOYEES.EMPLOYEE_ID =
DEPARTMENTS.MANAGER_ID
  WHERE DEPARTMENTS.DEPARTMENT_ID = v_dep_id;
  RETURN v_ln || ' '|| v_fn;
END FUNCTIONS03_1_GREILINGER;
```

2. Erstellen Sie eine Funktion, die die MitarbeiterNr bekommt und die Anzahl seiner Jobs liefert (2L)

```
create or replace FUNCTION ANZAHL_JOBS
(
  P_EMP IN employees.employee_id%type
) RETURN NUMBER AS
  v_anzahl_jobs number;
  v_emp         employees.employee_id%type;
BEGIN
  v_anzahl_jobs := 0;

  if (p_emp is NULL) then
    NULL;
  else

    /* hatte der Mitarbeiter bereits Jobwechsel? */
    select count(*)
    into   v_anzahl_jobs
    from job_history
    where employee_id = p_emp;

    select employee_id
    into   v_emp
    from employees
    where employee_id = p_emp;

    v_anzahl_jobs := v_anzahl_jobs +1;

    RETURN v_anzahl_jobs;
  end if;

Exception
  when NO_DATA_FOUND THEN
    RAISE_APPLICATION_ERROR(-20202, 'Employee '||p_emp||' does not
exist');
  when others then
```

```
    NULL;

END ANZAHL_JOBS;
```

```
CREATE OR REPLACE FUNCTION "5BHIF20167_04"."FUNCTIONS03_2_GREILINGER"
(
    emp_id IN INT
) RETURN VARCHAR2 AS
  v_emp_name employees.first_name%TYPE;
  v_numb INT;
BEGIN
  SELECT
    EMPLOYEES.FIRST_NAME,
    COUNT(JOB_HISTORY.EMPLOYEE_ID)
    INTO v_emp_name, v_numb
  FROM
    EMPLOYEES INNER JOIN JOB_HISTORY ON JOB_HISTORY.EMPLOYEE_ID =
EMPLOYEES.EMPLOYEE_ID
  WHERE
    EMPLOYEES.EMPLOYEE_ID = emp_id
  GROUP BY
    EMPLOYEES.FIRST_NAME;
  RETURN 'Employee ' || v_emp_name  || ' <id: ' || emp_id || '>
number of jobs: ' || v_numb;
END FUNCTIONS03_2_GREILINGER;
```

3. Erstellen Sie eine Funktion, der die ManagerID mitgegeben wird. Diese Funktion soll eine Liste aller seiner Mitarbeiter liefern. (1L)

```
CREATE OR REPLACE FUNCTION "5BHIF20167_04"."FUNCTIONS03_3_GREILINGER"
(
    v_man_id IN INT
) RETURN VARCHAR2 AS
  v_names VARCHAR2(200);
BEGIN
  v_names := 'employees for manager with id: ' || v_man_id || ':';
  FOR emp_name IN
    (
      SELECT
        EMPLOYEES.LAST_NAME
      FROM
        EMPLOYEES
      WHERE
        EMPLOYEES.MANAGER_ID = v_man_id
    )
    LOOP
      v_names := v_names || ' ' || emp_name.LAST_NAME;
    END LOOP;
      RETURN v_names;
END FUNCTIONS03_3_GREILINGER;
```

Übungszettel 4 – Exceptions HR

1. Ändern Sie das Gehalt des Mitarbeiters 129: (2L)

- auf das Gehalt des Mitarbeiters mit dem Vornamen 'Peter'.
- wenn 'Peter' nicht in der Datenbank gespeichert ist, dann ermitteln sie das Durchschnittsgehalt aller Mitarbeiter
- wenn es mehr als einen 'Peter' gibt, dann ermitteln Sie das kleinste Einkommen aller 'Peter'

Verwenden Sie Exceptions!

```
CREATE OR REPLACE PROCEDURE UE4_A1 AS

v_salary employees.salary%TYPE;

BEGIN
  select min(salary)
  into v_salary
  from EMPLOYEES
  where FIRST_NAME='Peter'
  group by first_name;

  EXCEPTION
  WHEN NO_DATA_FOUND THEN
      select avg(salary)
    into v_salary
    from EMPLOYEES;

  update EMPLOYEES
  set SALARY=v_salary
  where employee_id=1;
END UE4_A1;
```

```
set define off;


CREATE OR REPLACE PROCEDURE "5BHIF20167_04"."EXCEP_04_01" AS
  v_salary employees.salary%TYPE;
BEGIN
  v_salary := EXCEP_04_01_HELP('Peter');
  --DBMS_OUTPUT.PUT_LINE(v_salary);
  UPDATE employees
  SET salary = v_salary
  WHERE employee_id = 129;
END EXCEP_04_01;
```

Dazugehörige Funktion

```
CREATE OR REPLACE FUNCTION "5BHIF20167_04"."EXCEP_04_01_HELP"
(v_emp_fn IN EMPLOYEES.FIRST_NAME%TYPE)RETURN EMPLOYEES.SALARY%TYPE
AS
  v_salary EMPLOYEES.SALARY%TYPE;
BEGIN
  SELECT salary INTO v_salary
  FROM employees
```

```
  WHERE first_name = v_emp_fn;

  RETURN v_salary;
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    --DBMS_OUTPUT.PUT_LINE('no data');
    SELECT AVG(salary) INTO v_salary
    FROM employees;
  WHEN TOO_MANY_ROWS THEN
    --DBMS_OUTPUT.PUT_LINE('too many');
    SELECT MIN(salary) INTO v_salary
    FROM employees
    WHERE first_name = v_emp_fn;
    RETURN v_salary;
  WHEN OTHERS THEN
    --DBMS_OUTPUT.PUT_LINE('!WARNING! OTHER EXCEPTION');
    RETURN null;
END EXCEP_04_01_HELP;
```

2. Erstellen Sie eine Prozedur, der die Abteilungsnummer mitgegeben wird. Suchen Sie den Mitarbeiter mit dem höchsten Einkommen und machen Sie diesen zum Manager der Abteilung. (2L)

Programmieren Sie sicher und verwenden Sie Exceptions. Geben Sie Beispiele an, wo die Verarbeitung funktioniert und wo es Probleme gibt.

```
CREATE OR REPLACE PROCEDURE UE4_A2 AS

v_manager_id employees.employee_id%type;
v_manager_dep employees.department_id%type;

BEGIN
  select employee_id, department_id
  into v_manager_id, v_manager_dep
  from (
    select employee_id, department_id
    from EMPLOYEES
    where FIRST_NAME='Peter'
    order by salary)
  where ROWNUM = 1;


  update EMPLOYEES
  set MANAGER_ID=V_MANAGER_ID
  where department_id=v_manager_dep;

  EXCEPTION
  WHEN NO_DATA_FOUND THEN
  dbms_output.put_line('No Data was returned');
  RAISE;

  WHEN TOO_MANY_ROWS THEN
  dbms_output.put_line('Multiple Employees with the same salary has
been found');
  RAISE;
  WHEN others THEN
  dbms_output.put_line('An error occured');
  RAISE;
```

```
END UE4_A2;
```

```
set define off;

CREATE OR REPLACE PROCEDURE "5BHIF20167_04"."EXCEP_04_02"
(v_department_id IN departments.department_id%TYPE) AS
  v_man_id employees.employee_id%TYPE;
BEGIN
  SELECT manager_id INTO v_man_id
  FROM departments
  WHERE department_id = v_department_id;
  IF v_man_id IS NULL THEN
    UPDATE departments
    SET manager_id = (
      SELECT employee_id
      FROM employees
      WHERE salary = (
        SELECT MAX(salary)
        FROM employees
        WHERE department_id = v_department_id
      )
    )
    WHERE department_id = v_department_id;
  ELSE
    DBMS_OUTPUT.PUT_LINE('manager already set!');
  END IF;
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('department not found or without
employees!');
END EXCEP_04_02;
```

Übungszettel 5 – Trigger HR

1. Stellen Sie sicher, daß an der Mitarbeitertabelle vor 5:00 und nach 23:00 keine Änderungen gemacht werden können. (2L)

```
CREATE OR REPLACE TRIGGER UE5_A1
BEFORE INSERT OR UPDATE OR DELETE ON EMPLOYEES
DECLARE
    oowhe exception;
BEGIN

  IF CURRENT_TIMESTAMP  < '05:00' OR CURRENT_TIMESTAMP > '23:00' THEN
    RAISE oowhe;
  end if;

END;
```

```
CREATE OR REPLACE TRIGGER "5BHIF20167_04"."TRIGGER_5_1_GR"
BEFORE UPDATE OR INSERT OR DELETE ON employees
BEGIN
  IF EXTRACT(hour FROM SYSTIMESTAMP) > 23 OR
  EXTRACT(hour FROM SYSTIMESTAMP)< 5 THEN
    RAISE_APPLICATION_ERROR(-20201,'This employee can not be updated
during 23:00 and 5:00.');
  END IF;
END;
/
ALTER TRIGGER "5BHIF20167_04"."TRIGGER_5_1_GR" DISABLE;
```

2. Stellen Sie sicher, dass das Gehalt eines Mitarbeiters nicht verkleinert werden kann. (1L)

```
CREATE OR REPLACE TRIGGER "5BHIF20167_04"."TRIGGER_5_2_GR"
BEFORE UPDATE OF salary ON employees FOR EACH ROW
BEGIN
  IF :OLD.salary > :NEW.salary THEN
   RAISE_APPLICATION_ERROR(-20202,Gehalt geht nix kleiner!');
  END IF;
END;
/
ALTER TRIGGER "5BHIF20167_04"."TRIGGER_5_2_GR" DISABLE;
```

3. Schreiben Sie einen Trigger, der sicher stellt, dass der Mitarbeiter und der Vorgesetzte zur gleichen Abteilung gehören. (1L)

```
CREATE OR REPLACE TRIGGER "5BHIF20167_04"."TRIGGER_5_3_GR"
BEFORE INSERT OR UPDATE OF MANAGER_ID ON EMPLOYEES FOR EACH ROW
DECLARE
  v_dep_id DEPARTMENTS.DEPARTMENT_ID%TYPE;
  department_manager_mismatch EXCEPTION;
BEGIN
  -- change select depending on insert/update
  SELECT DEPARTMENT_ID INTO v_dep_id
  FROM employees
  WHERE employee_id = :new.manager_id;
-- nachlesen des gleichen datensatzes -> mutating trigger
```

```
  IF v_dep_id != :new.department_id THEN
    RAISE department_manager_mismatch;
  END IF;
EXCEPTION
  WHEN department_manager_mismatch THEN
    RAISE_APPLICATION_ERROR(-20203, 'employee department and manager
department mismatch');
END;
/
ALTER TRIGGER "5BHIF20167_04"."TRIGGER_5_3_GR" ENABLE;
```

4. Jedes Mal, wenn der Job eines Mitarbeiters geändert wird, soll eine Logging Information (job history) geschrieben werden: (1L)

- Employee ID, alte job ID, neue department ID, Einstellungsdatum, Austrittsdatum.

- wenn schon ein Logging Datensatz für Mitarbeiter vorhanden ist, dann soll das Eintrittsdatum auf Enddatum+1 gesetzt werden.

```
CREATE OR REPLACE TRIGGER "5BHIF20167_04"."TRIGGER_5_4_GR"
AFTER UPDATE OF JOB_ID ON EMPLOYEES FOR EACH ROW
DECLARE
  v_jobs number;
  v_end number;
BEGIN
  select count(*) into v_jobs
  from job_history
  where employee_id = :old.employee_id;
  if v_jobs > 0 then
    v_end := 1;
  elsif v_jobs = 0 then
    v_end := 0;
  end if;
  INSERT INTO JOB_HISTORY
  VALUES (:old.employee_id,SYSDATE,SYSDATE+v_end,:old.job_id,
:new.department_id);
END;
/
ALTER TRIGGER "5BHIF20167_04"."TRIGGER_5_4_GR" ENABLE;
```

PLÜFen vom letzten Schuljahr (Teil 1)

1. Implmentieren Sie einen Trigger, der sicher stellt, dass ein Mitarbeiter genau so viel verdient, wie das Max_Salary seiner Job_ID (insert)

2. Schreiben Sie einen Trigger, der die Email Adresse eines Mitarbeiters (beim Insert und falls leer) ändert auf <Vorname>,<Familienname>@<Department_Name>“

3. Erstellen Sie eine Prozedur, die überprüft ob das Gehalt eines Mitarbeiters zum Min- bzw. Max_Salary seiner Job_ID passt. Geben Sie dem Mitarbeiter eine Gehaltsreduktion um 5%, falls er mehr als das Max_Salary verdient.

4. Erstellen Sie eine Prozedur die einen neuen Mitarbeiter anlegen kann bzw. einen Jobwechsel eines bestehenden Mitarbeiters durchführen kann.

   Analysieren Sie die Tabellen employees und job_history. Sehen sie den Inhalt der Tabellen an!

PLÜFen vom letzten Schuljahr (Teil 2)

1. Implementieren Sie einen INSERT Trigger, der prüft ob MIN_Salary und MAX_Salary eines JOBS befüllt ist. Falls eines der beiden leer ist, setzten sie den vorhandenen Wert ein. Falls beide leer sind, erzeugen Sie eine entsprechende Exception.

```
create or replace TRIGGER ITR

BEFORE INSERT ON JOBS

for each row

BEGIN

 if (:new.MIN_SALARY IS NULL) AND (:new.MAX_SALARY IS NULL) then

 RAISE_APPLICATION_ERROR(-20000, 'Both min_sal and max_sal is null');

 elsif (:new.MIN_SALARY IS NULL) then

 :new.MIN_SALARY := :new.MAX_SALARY;

 elsif (:new.MAX_SALARY IS NULL) then

 :new.MAX_SALARY := :new.MIN_SALARY;

 end if;

END;
```

2. Erstellen Sie eine Prozedur, die überprüft ob das Gehalt eines Mitarbeiters grösser als das Durchschnittsgehalt seiner Kollegin in den anderen Abteilung ist. Geben Sie dem Mitarbeiter - in diesem Fall – eine Gehaltsreduktion um 5%

3. Erstellen Sie eine Funktion, die die Daten Location/Department/Employees korrekt anlegen kann.

4. Schreiben Sie eine Funktion, die für einen Mitarbeiter ermittelt, ob dieser einen Job Wechsel hatte. Falls das der Fall war, überprüfen Sie, ob sich der Job Title geändert hat. Geben Sie diese beiden Job Title durch „,“ getrennt von der Funktion zurück.

PLÜF von der AHIF

1. Implementieren Sie einen Trigger, der sicher stellt, dass ein Mitarbeiter genau so viel verdient, wie die Hälfte der Max_Salary seiner Job_ID (update)

2. Schreiben Sie eine Funktion, die die Email Adresse eines Mitarbeiters (falls leer) ändert auf <Vorname>@<Familienname>.<Department_Name>“

3. Erstellen Sie eine Prozedur, die den Gehalts- Mittelwert ((min+max)/2) einer gegebenen Job_ID ermittelt. Suchen Sie einen Mitarbeiter mit der gegebenen JOB_ID, dessen Salary möglichst gleich diesem Mittelwert ist.

4. Erstellen Sie eine Prozedur die einen neuen Mitarbeiter anlegen kann bzw. einen Jobwechsel eines bestehenden Mitarbeiters durchführen kann.