Given following relational schema:

- **flight**
  - (label, date, destination, flight_time, distance)

- **departure**
  - ((flight_label, flight_date) → flight, (type, serial_number) → plane, captain)

- **employee**
  - (ID, name, address, job, salary)

- **plane_type**
  - (type, manufacturer, number_of_seats, cruising_speed)

- **plane**
  - (type → plane_type, serial_number, purchase_date, flight_hours)

- **spare_part**
  - (ID, label, price)

- **requires**
  - (type → plane_type, serial_number → plane, spare_part_ID → spare_part)

- **pilot**
  - (employee_ID → employee, license, flight_hours)

- **technician**
  - (employee_ID → employee, team_number)

- **can_fly**
  - (employee_ID → pilot, type → plane_type)

- **can_maintain**
  - (employee_ID → technician, type → plane_type)

- **passenger**
  - (passenger_ID, name, address, age)

- **booking**
  - (passenger_ID → passenger, (flight_label, flight_date) → flight, class, seat_number, price)

1. Formulate following queries in SQL:
    
    (a) Query name and salary of all employees that earn more than 6450€.
    
    (b) Query name and salary of all employees that do not earn between 6000€ and 10000€.
    
    (c) Query all planes that are of type A−240 or TRIDENT. Sort by purchase date.
    
    (d) Which pilots have a license different from I and II.
    
    (e) Determine the names of employees that have an A at the third position of their name.
    
    (f) Determine the names of employees that have the L twice in their name.
    
    (g) Query name, job and salary of all employees whose job is either Dipl.–Ing. or steward/–ess and that earn are least 6000€.
2. Define following queries in SQL:

(a) Determine employee Id, name and salary of all employees. Add an intermediate column new_salary to the result that shows the current salary increased by 15%. The new_salary must be returned as integer value.

(b) Given your solution from task (a), add another intermediate column that shows the difference between the original salary and the new_salary. The difference must also be returned as integer value.

(c) For every plane, list its type, serial number and operating hours. Operating hours must be returned in a column called operating_hours.

(d) Create a query that returns following string for every employee:

< name > earns < salary > per month, but desires to earn < 3\times salary >.

Replace all placeholders with the respective data using SQL. The new column is called desired_salary.

(e) List all plane types. Thereby, all first letters must be capitalized, the rest must be uncapitalized. Return the length of the type name in a second column. The columns are called name und length.

3. Use SQL to retrieve following information:

(a) How many planes (not types) are stored in relation departure?

(b) Determine the number of employees that have a doctor’s degree (Dr. or PhD)?

(c) What is the average_salary by job?

(d) Return the total_price and the number_of_bookings for all journeys in 1993 in relation booking.

(e) Determine the minimum salary for every job!

(f) Retrieve the difference between the maximum and minimum salary of the all employees.
4. You are new in the university IT department. Your task is to reformulate following SQL query as it is to slow currently:

```sql
SELECT DISTINCT X.exam_ID FROM exams X
WHERE X.exam_ID IN (SELECT Y.exam_ID FROM exams Y
WHERE Y.student_ID <> X.student_ID);
```

Maybe, a reformulation will improve the query performance. In order to reformulate it, solve following tasks:

(a) What is the result of this query?
(b) You found two ways to reformulate the query. Reformulate the query
   i. Without using a nested query in the WHERE clause!
   ii. Without using tuple variables (to distinguish the use of the same relation), but by using aggregation functions.

5. Create the following views in SQL:

(a) i. Create a view `employee_view` on relation `employee` that only shows the name, the address and the job of all employees. Rename the columns `name` into `empview_name` and `address` into `empview_address`!
ii. Now, the view `employee_view` should only list employees which earn more than 5000€ .
iii. Is it possible to insert new employees into view `employee_view`?

(b) Given following relation:

```sql
exams (course_of_studies, course, student, examiner, date, mark)
```

Define following views using SQL:

i. The computer science faculty can only view data of students that are registered in `computerscience`.
ii. The examination office can view all data.
iii. The scholarship commission can only view average marks of every student.
iv. The dean can only view data about exams of the last year for statistical purposes (i.e., the relationship to students and examiners must be removed).

Good Luck!