University of Sussex Autumn 2017
Informatics

Databases
Assignment 2 (Deadline 7.12.17, 4pm)

This coursework must be submitted electronically via E-submission. You have to answer *all* 15 questions. Your answers must be uploaded in one file *using the template file* published on the StudyDirect page. This template contains essential comments to steer the automatic testing rig. Please follow the instructions carefully and do not remove any comment line from that template.
Don't write your name anywhere within the file but include your candidate number in the first line as a comment (e.g. `-- 30455`) just in case.

YOU MUST WORK ON THE ASSIGNMENT ON YOUR OWN! The standard rules for collusion and plagiarism apply and any cases discovered will be reported and investigated.

Detailed Instructions (follow carefully)
This assignment refers to an implementation of the firewood sales company database as designed in the the first assignment (see StudyDirect). To be able to answer the questions of this second assignment you must first run the SQL script `a2-setup.sql` that defines the tables that your code will rely on[1]. It is available from our StudyDirect site. For the completion of this assignment it will be necessary to inspect the table structure set up by this script. Do not modify the structure of the tables in the given script when you write your answers unless told to do so.
Note that a few data records have been inserted into the tables to help you test your answers. It is recommended that you test your code with additional sample data you insert into the tables yourself. However, *do not include any of the test data or the corresponding insert statements in your submission*. Also, you must *not* include the code of `a2-setup.sql` in your answer.
In the following, SQL always refers to the MySQL 5.1 dialect and all your code must run *on our ITS server* where it will be tested for marking purposes. Make sure you get the expected results on *on our ITS server*. If you

[1]Note that the implementation is a translation of the E/R model given in the model answers of Assignment 1, with a few minor deviations: address information has been moved up into the *FW_Person* schema. For salespeople we added a *bonusAmount* column and a *bonusDate* column. Their purpose will become clear further below.

test it on other servers you might get different behaviour. Don't forget that in MySQL table names are case sensitive.
You must not deviate from the requested order and name of the columns in queries. Note that for every query the names and order of columns is clearly specified in the paper. Any change will most likely confuse the testing script and cost you marks.
Each question does only allow one SQL statement as answer. Copy this statement *directly below* the corresponding question comment, e.g. `--@@01` if you answer Question 1, on a new line in the template file. Do not remove any of the template comments as they drive the testing rig. If you cannot answer a question just leave the space empty below the question number comment. You can submit partial answers but they must be syntax correct and run. The marks you get for those depends on the quality of the answer. Where a query is very complex, you might wish to add comments to help the marker appreciate what you have done in case your query is not correct.

1. Write SQL code to set up table `FW_Lorry` according to the following Relational Schema:

   FW_Lorry(vehicleRegNo,make,model,maxLoad,accessories)
   primary key vehicleRegNo

   You must accommodate the following requirements:

   (a) For table and column names you *must* pick the names used in the schema above (otherwise you will lose marks as tests will fail).

   (b) The vehicle registration number may contain letters and is *always* 7 characters long.

   (c) The make of the lorry is *always* one of the following: Volvo, Ashok, Ford, Hyundai, Iveco, MAN, Mercedes-Benz, Scania, Skoda, Tata. The company does not use any other. When ordered, they should be ordered alphabetically on their string values. Comparison, however, should be normal string comparison (like standard collation prescribes for strings).

   (d) The accessories can be any combination of the following four strings: Combination, Dual, EOBR, Tandem (which stand for combination truck, dual wheel usage on truck, electronic onboard recording system, and use of tandem axle on truck).

(e) The maximum load is measured in tons and never greater than 99.9 tons. We only keep one digit after the decimal point. The default load is 40.0 tons.

Your code *must* execute *without syntax error* and *without runtime error* assuming that all other tables have been set up by the script `a2-setup.sql` which you must not include. [7 marks]

2. Write SQL code to set up table `FW_TransportRequirement` according to the following Relational Schema:

   FW_TransportRequirement(number,order_no,lorry,transport_quantity)
   primary key (number,order_no)
   foreign key order_no references FW_SalesOrder(order_no)
   foreign key lorry references FW_Lorry(vehicleRegNo)

   You must accommodate the following requirements:

   (a) For table and column names you *must* pick the names used in the schema above (otherwise you will lose marks as tests will fail).

   (b) For each order there are never more than 255 transport requirements.

   (c) The type of the transport quantity column must be like the one for the maximum load for lorries given in the question 1e) above, however the default must now be 0.

   (d) When an order is deleted, all transportation requirement records for this order shall be deleted automatically as well.

   (e) When an order's order number changes (which will rarely happen), then the change is automatically and consistently performed in the affected transport requirements.

   (f) When a lorry is deleted from the database, the transport requirements affected should remain on the database (not affecting the deletion of the lorry otherwise).

   (g) If and where applicable equip the foreign constraints with constraint names of your choosing.

   Your code *must* execute *without syntax error* and *without runtime error* assuming that all other tables have been set up by the script `a2-setup.sql` which you must not include. [11 marks]

Additional Instructions (Queries) Questions 3–13
For each of the tasks specified below write *one single* SQL query, respectively, that solves the task. You can use nested queries (i.e. subselects) wherever you like. You must not use (nor declare) any stored procedures or functions or any tables for Questions 3–13. You *must produce column headings as specified with each query*. Do not change order or name of the columns as this will cause tests to fail which will cost you marks. *It is important that your queries will work correctly with* any *data (according to the schema). All references to time, when not explicit, are relative and refer to the time of running the query.*

3. Without recreating any tables, add to `FW_Person` a column `dob` (use the exact same spelling please) for their date of birth. [4 marks]

4. Remove leading `0`s from all house numbers in `FW_Person`. Instead of performing a complex string manipulation using single-row string functions, it may be easier to simply use the expression

   ```
   TRIM(LEADING '0' FROM ? )
   ```

   where you need to replace the question mark by the corresponding expression required for this question. [5 marks]

5. Remove all instances of the client phone number `01273007007` from the database. Clients who gave this number were joking. It is not a valid phone number. [4 marks]

6. List the person identifier of all those salespeople who managed to achieve the following feat at least once: sell more than 200 cubic metres in *one single order*. Sort the result table alphabetically. The headings must look like this:

   ```
   salesPerson_id
   ```

   [5 marks]

7. For all salespeople, list their person id, last name and monthly sales target. The table must be ordered by last name. Rows with equal last names must be ordered by monthly sales target with the highest listed first. The headings must look like this:

```
person_id   lastname   monthly_sales_target
```

[6 marks]

8. List how many orders have been placed in the last 20 days. Here "the last 20 days" is supposed to refer to the 20 days up to, and including the day the query is run. Your result table should contain one column only. The heading must look like this:

   ```
   number_orders
   ```

   [6 marks]

9. List all the clients whose first name starts with the same two letters as their last name. Letters with different capitalisation are to be considered different in this case, so `a` and `A` are *not* the same. The result table should include person id, their first and last name. The headings must look like this:

   ```
   person_id   firstname    lastname
   ```

   [6 marks]

10. For each order, compute its total order price which is the sum of the quantities in the order multiplied by the price per cubic metre (column `price` in `FW_SalesOrder`). For instance, order number `101` has a total price of `5055.00`. Your result table should contain two columns: the order number and the total price. The result table should be sorted by column `total` with the highest price appearing first (at the top). The headings must look exactly like this:

    ```
    order_no    total
    ```

    [6 marks]

11. For each salesperson, list their turnover for the *current* quarter. – i.e. their sales worth in pounds which is the sum of "volume $\times$ price" for each of their individual sales made in this quarter. This means in particular that in joint orders you should disregard the sales of other salespeople. The term "quarter" refers to the standard meaning in

business where a year is divided into four quarters (the first covering January to March, and so on).

Your result table should contain three columns: person identifier, their name as one column consisting of initial of first name followed by a space, followed by their last name, e.g. *N Nobody* (and not *Nelly Nobody*), and the total sales for the salesperson in the current quarter. The headings must look exactly like this:

```
person_id  name  quarterly_sales
```

Hint: You are expected to consult the MySQL manual here to find the right date functions that deal with quarters which we have not used so far. [8 marks]

12. For every salesperson compute the average percentage of their contributions of order *quantity* (i.e. volume). For instance, assume a salesperson has sold a quantity of 40 m^3 in a first order where the total overall quantity was 200 m^3, and that they sold twice 10 m^3 to two different customers, i.e. a quantity of 10 + 10 = 20 m^3 in a second order with 20 m^3 being the order's total. Assume further that this salesperson was not involved in any other order. Then their average percentage would be calculated like so:

    $(40/200 + 20/20)/2 \times 100 = 0.6 \times 100 = 60.$

    Note that the average to be computed is the average of percentage points, not the average of quantities themselves. Don't round any intermediate results but the final percentage should be rounded to an integer value using MySQL function `ROUND`. Salespeople who have not sold anything should not appear in the list. The headings must look exactly like this:

    ```
    person_id  avg_vol
    ```

    [6 marks]

13. The quantity sold to one client in one order shall be called "individual sales quantity". Two (different) salespeople are called "co-sellers" if they made *both* sales for *one and the same* order (in the sample data provided `UYT00158U` and `ZAS10158U` are co-sellers for example). Note that nobody is considered to be co-seller of themselves. Find all

salespeople whose average individual sales quantity *per order* (considering only orders they have been selling for) is greater than the maximum individual sales quantity of all of their co-sellers.

The result table should have three columns: the person identifier of the salesperson, their average individual sales quantity, and the maximum individual sales quantity of the respective co-sellers. The headings must look exactly like this:

```
id   avg_sales  max_cosales
```

[9 marks]

Additional Instructions (Stored Procedures) Questions 14–15
For developing answers to Questions 14–15 you can use any delimiter you like. But do *not* declare the delimiter in the submission file and *remove the delimiter symbol from the end of each routine declaration* in the submission file. This is important for the automatic testing.

Note that successfully declaring a stored procedure does not necessarily mean it runs without error. You need to run and test your procedures to ensure that. Strictly name the stored procedures as indicated in the question. You are not allowed to include any other stored routine definitions, but you can use the answer from Question 14 in your answer to Question 15.

14. Write a stored function `individualSalesUpTo` that, given a salesperson's identifier as first argument and two dates as second and third argument, respectively, returns as `INTEGER` the turnover (i.e. sales worth in pounds) of the specified salesperson between the specified first and second date, including the sales on both dates. If the specified salesperson does not exist the function must return `-1`. If the salesperson does exist but any of the two specified dates is not a valid date the function must return `-2`.

    Illegal dates include dates that do not follow the standard MySQL date format, dates with incorrect month or day values, e.g. `2017-12-88`, dates where years are larger than `9999`, and in particular `0000-00-00`. If the second date is actually *before* the first date, the turnover to be returned is simply 0. [7 marks]

15. Write a stored procedure `setBonuses` that, given a date as argument sets up the bonuses for all salespeople for the month of the given date as explained below. For instance, if the given date is `2016-09-08` then the bonuses are to be computed for September 2016. If the date is illegal the procedure should abort with an error and error message:

    `PROCEDURE` *`accountname`*`.date does not exist`

    where *accountname* is the name of the ITS account running the procedure, so when *you* run it this will be *your* account name.

    Illegal dates include dates that do not follow the standard MySQL date format, dates with incorrect month or day values, e.g. `2017-12-88`, dates where years are larger than `9999`, and in particular `0000-00-00`. Please ensure that the execution is actually giving an error and that the error message is exactly as described here (otherwise testing will fail and you will lose marks). Note that you must not attempt to produce a string on the screen using a `SELECT` statement. In order to create those error messages, please consult Lecture 17, which explains how an error can be raised (simulated) in MySQL 5.1.

    The bonuses for a particular month are set up as follows: if the monthly target of a salesperson is met or surpassed by their individual sales for that month and their bonus amount is currently zero then their bonus amount is set to 10 per cent of the difference between actual sales for this month and target sales. This difference is then rounded down to the nearest integer. If their current bonus is not zero then their bonus amount is set to 15 per cent of the difference between actual sales for this month and the target sales. Again, this difference is rounded down to the nearest integer.

    If the monthly target of a salesperson is met or surpassed by their individual sales for the given month, the bonus date is set to the last day of this month. *Hint:* Use MySQL function `LAST_DAY`.

    If the monthly sales target has not been met the bonus amount is reset to zero. The bonus date remains unchanged in that case.

    There is one more requirement you need to implement (in cases where the procedure does not throw an error already): If there is any bonus date recorded in `FW_SalesPerson` that is already in the month passed to the procedure as argument, or if there is a bonus date even *later* than that argument, the procedure should do nothing at all (in

particular, it must not throw any error either). In this case one does know the procedure will have been run already successfully before.

[10 marks]