# Assignment III

## 1 Part one – SQL: Queries and Views

### *1.1 Objectives*

The purpose of this exercise is to practice writing queries in SQL, including the use of aggregate functions and views.

### *1.2 Background reading*

- SQL chapter(s) in your database book
- MIMER SQL documentation

### *1.3 The lab*

Use SQL to find the answers to the questions below towards your Jonson Brothers company database. Whenever a question requests information about entities that have both a number and a name, select both the number and the name to make your results more useful. For example, in question 3) return both the number and the name of the parts we are interested in.

1) List all employees, i.e. all tuples in the EMPLOYEE relation.
2) List the name of all departments, i.e. the NAME attribute for all tuples in the DEPT relation.
3) What parts are not in store, i.e. QOH=0? (QOH = Quantity On Hand).
4) Which employees have a salary between 9000 and 10000 (inclusive)?
5) What was the age of each employee when they started working (STARTAGE) here?
6) Which employees have a last name ending with “son”? Retrieve employee names and numbers.
7) Which items have been delivered by a supplier called “Playskool”? Formulate this query using a subquery in the where-clause.
8) Formulate the same query as above, but without a subquery.
9) What is the name and color of the parts that are heavier than a black tape drive? Formulate this query using a subquery in the where-clause. (The SQL query should not contain the weight as a constant.)
10) Formulate the same query as above, but without a subquery. (The query should not contain the weight as a constant.)
11) What is the average weight of black parts?
12) What is the total weight of all parts that each supplier in Massachusetts (“Mass”) has delivered? Retrieve the total weight for each of these suppliers.
13) Create a new relation (a table) that contains the items that cost less than the average price for items!
14) Create a view that contains the items that cost less than the average price for items! What is the difference between (13) and (14)?

15) Create a view that calculates the total cost of each sale, by considering price and quantity of each bought item. (To be used for charging customer accounts). Should return sale number and total cost. (Be aware that each sale could contain several rows in the table SALE.)

16) Suddenly, an earthquake strikes. Try to remove all suppliers in Los Angeles from the table SUPPLIERS. What happens? Why?

17) A database manager in the company has tried to find out what suppliers have delivered items that have been sold. He has created a help view and can find how many items are sold from each supplier of the items:

```
1 SQL> create view sale_supply(supplier, item, quantity) as
       select supplier.name, item.name, sale.quantity
       from supplier, item, sale
       where supplier.number = item.supplier and
             sale.item = item.number;
Ok
2 SQL> select supplier,sum(quantity) from sale_supply
       group by supplier;
6 rows selected
---------------------------------
|      SUPPLIER      |    SUM    |
---------------------------------
|       Cannon       |     6     |
|       Koret        |     1     |
|    Levi-Strauss    |     1     |
|     Playskool      |     2     |
|     White Stag     |     4     |
|     Whitman's      |     2     |
---------------------------------
```

Now he would also like to find out suppliers that have not had any of their delivered items sold. Help him! Drop and redefine sale_supply view in such a way that suppliers, that have delivered items that have never been sold, are considered as well.
**Hint**: The above definition of sale_supply uses an (implicit) inner join that removes suppliers that have not had any of their delivered items sold.

# 2 Part two – Database Integrity: Assertions, Triggers, and Stored Procedures

## *2.1 Objectives*

The purpose of this exercise is to give a good understanding of how to use SQL assertion checks, triggers, and stored procedures to maintain database integrity and to support applications.

## *2.2 Preparations*

Write your solutions on paper before testing them out on your database.
**Note**: Every procedure, trigger and function must be enclosed between two '@' characters, each on a separate line. As an example see the 'to_date' function defined in the 'company_data.sql' file.

## *2.3 Background reading*

- Chapter 9 and Section 24.1.5 in the course book
- MIMER SQL documentation

### *2.4 The lab*

1) Create an integrity constraint (by the alter table command) that ensures that no customer account can have a credit above 10000.

2) Create a stored procedure for creating customers. Should take name, street address, and city as arguments. Use a sequence for automatically generating a unique key.

3) Create a stored procedure that creates a customer account. Should take customer number and any credit limit (value = 0 if no credit) as arguments. Use a sequence for automatically generating a unique key.

4) Create a stored procedure for depositing money into a customer's account. Should take an account number and a positive amount as arguments. If amount is negative the deposit should be aborted.

5) Create a stored procedure that finalizes a sale by reducing qoh of the sold items and by charging a customer account the total cost of the sale. Should take the sale number and a customer account as arguments.

   **Hint**: Create some local variables to store intermediate values. Use the view created in lab exercise 1.3.15) for the sale total amount and use a cursor to iterate on the sold items.

6) Create an update trigger that checks if a customer account exceeds its given credit limit. If so, the trigger stores the date, account number, and the amount of the overdraft in a table (a new table created by you). If the credit limit is exceeded by more than 10% the transaction should be aborted. (By throwing an exception before the update is allowed to take place).

## 3 Handing in

Fill in your results in the **`dbt-3-template.sql`** file. Execute the commands in the resulting SQL file which will generate a **`.log`** file. Add to the .log file the explanations required and submit it in printed form.