#### OOK Investors Society

OOK Investors Society, a small financial club of which you are an investor, is considering automating some of their operations. You have volunteered to set up a database to help the investors track their portfolios.

The following is a brief description of the design specifications and business rules.

# Investments

Investments that are tracked will be limited to stocks only (no mutual funds, commodities, etc.). It has been decided that the system will track only the actual purchases and sales. The following form will be used to help collect data. Each day an investor can submit one or more forms that will indicate their transactions. Stored transactions can later be retrieved and reported on by investor, portfolio, exchange, broker, stock or date.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **DAILY STOCK TRADING RECORD** | | | | | | |
| Investor Name: John Smith Number: 123432  Address: 1567 - 18 Avenue Home Phone: 780-471-6677  City, Province: Edmonton, AB Postal Code: T5F 6Y7  Email: jsmith@dollars.com | | | | | | |
| **Stock** | **Exchange** | **Broker Name** | **Portfolio Name** | **Action** | **Quantity** | **Price/Share** |
| IBM | TSX | Mark Heatley | RSP | Buy | 50 | $ 82.90 |
| IBM | NYSE | Charlie Parker | Non-RSP | Buy | 150 | $ 83.25 |
| PGR | TSX | William Cuthbert | Spousal RSP | Sell | 20 | $ 42.40 |
| ORA | ASX | William Cuthbert | Speculative | Sell | 300 | $ 112.90 |
| Transaction Date: Jan 06, 2020 11:37 am | | | | | | |

# Additional Information

* A investor may have several transactions in one day
* A investor may trade the same stock only once per calendar day
* A stock can be traded on more than one exchange
* Brokers can trade on more than one exchange
* Brokers are paid a salary by the society so there are no commissions involved in any transactions
* The price per share, which may change during the day, is recorded at the time of each transaction

**Broker Registration**

The brokers are registered using the following form. A broker is only registered once, but can trade on multiple exchanges. A unique broker number will automatically be assigned when you insert a new broker into the database.

|  |
| --- |
| **New Broker Registration Form** |
| Broker Name: Alfred Jung Phone: 780-555-5877  Email: [firstpik@usa.com](mailto:firstpik@usa.com) |

## **Lab Questions**

**Part A (Due January 24, 2020 A01: 01:30 pm E01 06:50 pm)**

1. Create the following tables and corresponding constraints using **only** the Create table statement. Ensure that table and column names are **identical** to what is specified below and do **NOT** use double quotes around your table or column names. **All** constraints must be **explicitly** named, and you must be consistent with your constraint names. **20 marks**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Account | **Account\_Number** | **PK** | Number (7,0) | Not Null |
|  | Account\_Balance |  | Number (13,2) | Not Null |
|  |  |  |  |  |
| Broker | **Broker\_Number** | **PK** | Number (7,0) | Not Null |
|  | First\_Name |  | Varchar2 (25) | Not Null |
|  | Last\_Name |  | Varchar2 (30) | Not Null |
|  | Area\_Code |  | Number (3,0) | Null |
|  | Phone\_Number |  | Number (7,0) | Null |
|  | Email\_Address |  | Varchar2 (50) | Not Null |
|  |  |  |  |  |
| Exchange | **Exchange\_Code** | **PK** | Varchar2 (4) | Not Null |
|  | Description |  | Varchar2 (50) | Not Null |
|  | Country |  | Varchar2 (20) | Null |
|  |  |  |  |  |
| Exchange\_Stock | ***Exchange\_Code*** | **PK** | Varchar2 (4) FK | Not Null |
|  | ***Stock\_Code*** | Varchar2 (10) FK | Not Null |
|  | Active\_YN |  | Char (1) | Not Null |
|  |  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Investor | **Investor\_Number** | **PK** | Number (7,0) | Not Null |
|  | First\_Name |  | Varchar2 (25) | Not Null |
|  | Last\_Name |  | Varchar2 (30) | Not Null |
|  | Street\_Address |  | Varchar2 (35) | Null |
|  | City |  | Varchar2 (25) | Null |
|  | Province |  | Char (2) | Null |
|  | Postal\_Code |  | Varchar2 (7) | Null |
|  | Area\_Code |  | Number (3,0) | Not Null |
|  | Phone\_Number |  | Number (7,0) | Not Null |
|  | Email\_Address |  | Varchar2 (50) | Not Null |
|  | *Account\_Number* |  | Number (7,0)FK | Not Null |
|  |  |  |  |  |
| Portfolio | **Portfolio\_Number** | **PK** | Number (7,0) | Not Null |
|  | *Investor\_Number* |  | Number (7,0)FK | Not Null |
|  | Portfolio\_Description |  | Varchar2 (50) | Null |
|  |  |  |  |  |
| Stock | **Stock\_Code** | **PK** | Varchar2 (10) | Not Null |
|  | Stock\_Name |  | Varchar2 (50) | Not Null |
|  |  |  |  |  |
| Transaction | ***Portfolio\_Number*** | **PK** | Number (7,0)FK | Not Null |
|  | ***Stock\_Code*** | Varchar2 (10)FK | Not Null |
|  | **Transaction\_Date** | Date | Not Null |
|  | *Exchange\_Code* |  | Varchar2 (4)FK | Not Null |
|  | *Broker\_Number* |  | Number (7,0)FK | Not Null |
|  | Buy\_Sell |  | Char (1) | Not Null |
|  | Quantity |  | Number (7,0) | Not Null |
|  | Price\_Per\_Share |  | Number (6,2) | Not Null |
|  |  |  |  |  |

Notes:

* Primary keys are **bolded** and are denoted by PK
* Foreign keys are *italicized* and are denoted by FK
* Area codes must be three (3) digits, default to 780 and cannot start with 0
* Provinces must be 2 upper-case alphabetic characters
* Postal codes must be either Canadian style (ZnZ nZn) or the 5-digit American style (mnnnn) where Z is an upper case alphabetic character, m is a number from 1 to 9 and n is a number from 0 to 9
* Email addresses must contain the @ sign character
* Active\_YN must only accept the upper case values ‘Y’ (Yes) or ’N’ (No)
* All monetary values must be greater than or equal to zero
* The transaction date will default to the current date
* Buy\_Sell must only accept the upper case values ‘B’ (Buy) or ’S’ (Sell)
* Quantity must be greater than zero

You can load your tables with the “Data File” which is/will be available from Moodle or my staff web site

1. Create the following sequences in your database. Ensure that the sequences are named **identical** to the specification and additional numbers are not pre-allocated. **1 Mark**

|  |  |  |
| --- | --- | --- |
| Sequence Name | Initial Value | Increment |
| Account\_Seq | 7550 | 50 |
| Broker\_Seq | 20 | 2 |
| Investor\_Seq | 1020 | 5 |
| Portfolio\_Seq | 560 | 10 |

**Part A Lab Submission is to include the following:**

* Ensure your tables and sequences are created within the Oracle DMIT environment (not just on your laptop or home computer)
  + If you have trouble with a particular constraint, **please** contact your instructor
  + Please do not change anything with respect to your tables and sequences after the due date/time until your lab has been **marked and returned**
* A well-organized, readable report, including a title page showing your name and Oracle login in a swing-clip folder
  + Your name and Oracle login **must** be visible from the outside of your folder
  + Three ringer binders and duo-tangs will **NOT** be accepted!
* A single-sided printout of your source code for Questions 1 and 2
* A single-sided printed discussion about part A of the lab including:
  + How long the lab took to complete (split between in class and out of class time)
  + Any significant problems that were encountered
  + Whether or not there was enough material provided in the lectures to complete the lab
  + What, if any, changes or improvements could be made to the lab
* One electronic non-compressed text file named Lab1A\_YourLastName\_YourFirstName.sql
* If you need to re-submit your lab after you have submitted but prior to the due date/time, email your instructor to adjust your submission to Draft Mode, after which you can submit the revised file via Moodle
* Your printed copy **MUST** match your electronic copy and database objects
* Do not include your password, the lab specification or the “Data File” in your lab submission
* The printed copy of your script file will be submitted to a drop box in W309 (A01) or to your instructor (E01)..
* The electronic copy of your script file will be submitted to Moodle.
* Any additional requirements as specified by your instructor

**Part B (Due February 28, 2020 A01: 01:30 pm E01 06:50 pm)**

1. Write a **procedure** (PR\_Q3) that will be used to change **all** the broker e-mail addresses with a given domain name to a new domain name. The procedure will be passed two strings: i) the original domain name that is to be replaced; and ii) the new domain name that will be replacing the original (in that order). For example, if the procedure is passed the strings "snailmail.com” and "slowpoke.com", every e-mail in the database with "snailmail.com" as their domain name will be changed to "slowpoke.com" as their domain name (e.g. "billy@snailmail.com” will become “billy@slowpoke.com”). The procedure must ensure that all variations (upper or lower case) of the first string are found and replaced by the second string. Update only those records that need to be updated and only update each record once. Use at least one **explicit** cursor in your solution.

For testing purposes, you will need to add "authid current\_user" to your procedure.

**Procedure Code 6 Marks, Test Case 2 Marks**

Your test case will consist of three steps

1. A select **and** display of the **appropriate** data from the **appropriate** table(s) **before** the procedure is executed
2. Execute your procedure (ensure your execute statement is included in your lab submission)
3. A select **and** display of the **appropriate** data from the **appropriate** table(s) **after** the procedure is executed
4. Write an overloaded **function** (FN\_Q4) inside a **package** (PKG\_Q4). This function is passed either: i) a portfolio number and a stock code (in that order); or ii) a portfolio description, an investor number and a stock name (in that order). You can assume that the portfolio and investor input parameters **are valid**. The function will return a string indicating the weighted average cost of the given stock for the given portfolio. If the stock information is not valid, return an appropriate error message including the invalid input parameter. If the portfolio **does** **not** contain the requested stock, the function will return the string "[investor first name] [investor last name] does not have any of the units of the stock [stock name] in the portfolio [portfolio description]”. If the portfolio **does** contain the stock, the function will return the string "[investor first name] [investor last name] paid, on average, [weighted average] for the stock [stock name] in the portfolio [portfolio description]”. All input parameters will be passed to the function in the correct order. Use **exception handling** in your solution. For testing purposes, you will need to add "authid current\_user" to your package specification only.

**Note:** Each function needs to stand alone within the package (i.e. one function must **NOT** call the other function).

**Function/Package Code 10 Marks, Test Cases 4 Marks**

Test Cases: Demonstrate your function with the following scenarios using both sets of parameters:

* 1. The stock name or stock code does not exist
  2. The stock name is not unique (parameter set “ii” only)
  3. The stock information is valid but the portfolio / stock combination does not exist
  4. The stock information is valid and the portfolio / stock combination does exist

Each test case will consist of two steps:

1. A select **and** display of the **appropriate** data from the **appropriate** table(s)
2. Call your function and display the result (ensure your function call statement is included in your lab submission).
3. Write a **trigger** (TR\_Q5) that will ensure the buying and selling of stocks (i.e. insertions into the Transaction table) follows two rules:
4. If the transaction is a "sell", the investor must currently own the quantity of the stock that they are trying to sell
5. If the transaction is a "buy", the investor must have enough money to make the purchase

If either of the rules is violated, the user should be notified with an appropriate error message and the processing should be discontinued. Otherwise, update the amount of money in the investor’s account accordingly.

**Trigger Code 6 marks, Test Cases 2 marks**

**CREATE OR REPLACE TRIGGER TR\_Q5**

**BEFORE INSERT or UPDATE OF \*\*\*look for relationship between buy &sell with another table in the database**

**ON TRANSACTION**

**For each row**

**Begin**

**If**

**Raise\_Application\_Error(-20099, ‘……’);**

Your test cases will display **both** successful and unsuccessful transactions for both rule “a” and “b”. Each test case will contain three steps:

1. A select **and** display of the **appropriate** data from the **appropriate** table(s) **before** the statement is executed
2. Execute your DML statement (ensure your DML statement is included in your lab submission)
3. Display of the error message (if applicable) **and** a select and display of the **appropriate** data from the **appropriate** table(s) **after** the statement is executed

***Part B Lab Submission* *is to include the following:***

* Ensure your procedure, package and trigger are created within the Oracle DMIT environment (not just on your laptop or home computer)
  + If you have trouble with a particular question, **please** contact your instructor
  + Please do not change anything with respect to your procedure, package and trigger after the due date/time until your lab has been **marked and returned**
* A well-organized readable report, including a title page showing your name and Oracle login in a swing-clip folder
  + Your name and Oracle login **must** be visible from the outside of your folder
  + Three ring binders and duo-tangs will be **NOT** accepted!
* A single-sided printout of your source code for Questions 3, 4 and 5
* A single-sided printout of your test cases (DML statements, Select statements and results) for Questions 3, 4 and 5
* Your printouts **MUST** match the corresponding electronic files and database objects
* A short single-sided printed discussion about the lab including:
  + What you liked and/or disliked about the lab
  + How long the lab took to complete (split between in class and out of class time)
  + Any significant problems that were encountered
  + Whether or not there was enough material provided in the lectures to complete the lab
  + What, if any, changes or improvements could be made to the lab
* Two electronic files:
  + Your source code as a single text file named Lab1B\_Source\_LastName\_FirstName.sql
  + Your test cases as a single text file named Lab1B\_TestCases\_LastName\_FirstName.sql
  + Compressed files (i.e. zip, rar, etc.) or folders will NOT be accepted
* If you need to re-submit your lab after you have submitted but prior to the due date/time, email your instructor to adjust your submission to Draft Mode, after which you can submit the revised file via Moodle
* Ensure there are no rollback, or commit, statements in your code!
* Do not include your password, the lab specification or the “Data File” in your lab submission
* **Select \* is NOT acceptable anywhere in your lab (including test cases and/or commented out code)**
* The printed copy of your script file will be submitted to a drop box in W309 (A01) or to your instructor (E01).
* The electronic copy of your script file will be submitted to Moodle.
* Any additional requirements as specified by your instructor

***Lab Expectations:***

* All code **MUST** be executable within the Oracle DMIT environment
* Tables, columns, sequences, procedures, functions, packages and triggers **MUST** be named exactly as indicated
* All cursors must be explicitly declared, opened, retrieved from and closed
* All code will be documented, where appropriate, and will follow a convention such as standards.doc (available from Moodle)
* Anything additional as specified by your instructor