Application Support: PL/SQL

Project: Process Trade Orders

# Course and Project Overview

This course and project consists of several parts.

## Part 0

* Investigate the components of PL/SQL and give a brief write-up.
* Part 0 is due by the end of day 0.
* Part 0 is NOT FOR MARKS.
* Feel free to use the Internet and any other resources.
* While you are working on part 1, your trainer will look over your answers and provide feedback where necessary to ensure you have a correct understanding of the components of PL/SQL.

## Part 1

Part 1 is itself divided into subsections, and you will hand in your answers for each section at different times, but in order to complete each section you must understand how all parts of the project fit together. Therefore, it is necessary that you read all subsections of Part 1 before beginning.

### Part 1a

* You will provide written answers to questions.
* You are encouraged to discuss your answers with your fellow trainees, but please do not allow other trainees to see your written work, or email your answers to each other. Each person’s written work should be their own.
* Part 1a is due by the end of day 3.
* While you are working on part 1b, your trainer will mark your answers for Part 1a and provide feedback.
* After receiving feedback, you should expect to make revisions.

### Part 1b

* You will write some PL/SQL stored procedures.
* Part 1b is due by the end of day 4.
* While you are working on Part 2, your trainer will mark your answers for Part 1b.

## Part 2

* You will need to investigate some data on an Oracle database.
* Your investigations and write-up should take only about 1 or 2 hours. The rest of your time on day 5 will be spent making revisions to Parts 1a and 1b.

# Part 0 – Investigate PL/SQL

As Application Support specialists working with PL/SQL, you may be required to write some simple stored procedures and functions but you will have to *support* the code that has been written by database developers. That is, you must be able to understand code when you see it.

For the following questions, provide a simple, short and clear explanation. Whenever possible, use the resources in the PLSQL Course Documentation on the LMS.

1. What is a PL/SQL basic block?
2. What are scalar variables? What data types can be used for scalar variables in PL/SQL? How can you declare local variables?
3. The CREATE GLOBAL TEMPORARY TABLE command creates global temporary tables (GTTs). Explain how these GTTs differ from regular tables.   
   (Please note: We want you to research the “CREATE GLOBAL TEMPORARY TABLE” command, not the “DECLARE GLOBAL TEMPORARY TABLE” command.)
4. Inside a PL/SQL block, you can declare data types called “records” which are complex or composite data types. What is the different between a “composite data type” and a “scalar data type”? How do you declare a record data type? How can you assign values to the components of a record?
5. PL/SQL can define “tables” which could be “associative array”, “nested table” and “varray”. Briefly explain each one.
6. What are database cursors? Explain “implicit cursors”, “explicit cursors”, cursors that are used within for loops, and cursors that are used with OPEN, FETCH and CLOSE. Check the file called **Database Code/Cursor1\_Implicit\_To\_Explicit.sql** to see examples ranging from the “most implicit” to “most explicit”.
7. What is “exception handling”? How does the code handle when errors occur?
8. In Oracle, what are a “procedures” and “functions”? How do you call procedures and functions? What are input parameters? How can you use output parameters?
9. What is a PL/SQL package?
10. In Oracle, what is a “trigger”? Explain the difference between “BEFORE”, “AFTER” and “INSTEAD OF” triggers. Explain the purpose of “:new” and “:old” inside trigger code.  
    Look at the file: **Final Project/TradeOrder\_ShareholderView.sql**   
    Please describe the purpose and usage of the SHAREHOLDER\_VIEW and the SHAREHOLDER\_INSTEAD trigger.
11. What is a "SAVEPOINT"?
12. Explain “READ COMMITTED”, and “SERIALIZABLE”, and “deadlock”.

# Part 1 - Process Trade Orders

Our company, **Another Friendly Exchange (AFE)**, hired FlyByNight Consulting to make some enhancements to our stock market trading platform. The database platform stores information for trades from exchanges around the world, but it was expanded to serve as the backbone for the "AFE Stock Exchange".

AFE is an electronic stock exchange that runs completely autonomously and without brokers. When shareholders place sell orders and buy orders, the system automatically matches these orders in order to allow trading to occur. The system handles market orders, limit orders, stop-loss orders and stop-limit orders. (It does not handle trailing-stop.) It also handles orders that are "day only", "good until canceled", "fill or kill", and "immediate or cancel".

For the foreseeable future, AFE Stock Exchange will be the only exchange which uses this automated matching system. However, other exchanges might someday adopt our order matching algorithm.

FlyByNight Consulting has completed the major part of the enhancements and the system demonstration was a complete success. The system has been installed on our Oracle Server. Unfortunately the system documentation is, shall we say, somewhat lacking. (Strictly speaking, documentation is completely lacking but the system does rather extensive tracing/debugging statements.) When we asked "Can we at least have the new ER diagram?" we were told "Oh yeah. We did make a few changes. We will track down an ERD for you.” Moreover, FlyByNight has now gone out of business and is not available for comment.

We have hired FDM to complete the work that was started by FlyByNight. We need you to create a new table and two more stored procedures. You will also update our data dictionary and write some entries into our knowledgebase to help our application support staff with the ongoing support of the system.

It is strongly recommended that you **read the entire specification** (this document) and **plan how you will proceed with the work**.

## Part 1a: Scripts From FlyByNight and Existing Documentation

FlyByNight provided us with three scripts:

**TradeOrder\_Schema.sql**

**TradeOrder\_PKG.sql**

**TradeOrder\_Testing.sql**

We also have the original data dictionary which FlyByNight did not update.

**TradeOrder\_DataDictionary.docx**

Investigate everything as you attempt to answer the following questions. Please provide answers that would help system operators and programmers understand our system. Your answers will form the basis for our knowledge base.

1. Our Java programmers will create a user-friendly web-based application but our programmers have little understanding of trading. We will need you to explain “market orders”, “limit orders”, “stop loss orders”, and “stop-limit orders”.
2. We need you to update our data dictionary. We want our data dictionary to have clear, concise (and useful) explanations for each table and column. Be sure to fully explain those items which require explanation, and keep the descriptions of items which are self-explanatory to a minimum.
3. Our Java programmers need to know how to enter “DAY ONLY”, “GOOD UNTIL CANCELLED”, “IMMEDIATE OR CANCEL” and “FILL OR KILL” orders into our system. Explain what each type of order is and how it is entered.
4. If “DAY ONLY” orders are not filled during the day, what happens to the orders? Does our system cancel the orders at the end of the day?
5. Investigate the code for the orders\_pkg and explain the algorithm which determines the value for the SHARE\_PRICE column in the TRADE table. Please trace through the algorithm all the way from the start (where the market price is retrieved from some table), to the middle (where the price might be adjusted), to the end (where the price is stored in the TRADE table).
6. During your investigation, did you discover any problems with our system?  
   N.B. In our company we try to adhere to the doctrine of “Completed Staff Work”. That is, do not simply report problems. For each problem, we want you to list options and a supply a recommendation.

"The Thinly Disguised Plot" – This is your opportunity to show that you understand SQL and PL/SQL code when you see it and can communicate technical ideas in a written report.

## Part 1b: Your Code

### New Table

We need a new table: STOCK\_HISTORY. Write the CREATE TABLE statement to create it.

**Purpose:** Allows storage and retrieval of stock history

**How STOCK\_HISTORY is used:** Users will SELECT, INSERT, UPDATE and DELETE stock history.

**When STOCK\_HISTORY is used:** Data is inserted at the opening of the day by the Open\_Trading\_Day procedure and updated at the end of the trading day by the Close\_Trading\_Day procedure. Stock\_History could be queried at any time and used for data analysis. (Data analysis is outside the scope of this project.)

|  |  |  |  |
| --- | --- | --- | --- |
| **STOCK\_HISTORY**  PRIMARY KEY (stock\_ex\_id, stock\_id, time\_open) | | | |
| Stock\_ex\_id | NUMBER(6) | NOT NULL | ID for the Stock Exchange |
| Stock\_id | NUMBER(6) | NOT NULL | ID for the Stock |
| Trading\_Date | DATE (time truncated) | NOT NULL | Date (without time) which identifies the trading day. Database must enforce that the time is truncated. |
| Time\_Open | DATE | NOT NULL | Date and time when the trading of the day started |
| Time\_Close | DATE | NULL | Date and time when the trading of the day ended |
| Open | NUMBER(10,4) | NULL | Opening price. Null If the stock has never been traded on the exchange. |
| High | NUMBER(10,4) | NULL | High Price for the day. Null if the stock was not traded on the day. |
| Low | NUMBER(10,4) | NULL | Lowest Price for the day. Null if the stock was not traded on the day. |
| Close | NUMBER(10,4) | NULL | Price at closing time. Null if the stock has never been traded on the exchange |
| Volume | NUMBER(12,4) | NULL | Total shares traded during the day. (Ex. If only two trades occur and 100 shares change hands in one trade, and 200 shares change hands in the other trade then the total volume for the day is 300.) |

### New Procedures

FlyByNight did not complete all of the programming that we require. In this part of your assignment, we have provided a description of the procedures that you need to write.

#### Open\_Trading\_Day Procedure

**Purpose**: Prepares for the trading day. It records the opening price for all stocks traded on the Exchange in the STOCK\_HISTORY table.

**When Open\_Trading\_Day is Used:** Open\_Trading\_Day must be called as close as possible to the time when the exchange opens. It records the opening time for the exchange and uses SYSDATE. If we do not run Open\_Trading\_Day close to the time of the actual opening then we will have to manually update the opening time. It would be best if Open\_Trading\_Day were scheduled to run automatically but it is beyond the scope of this course to use the job scheduler.

**How Open\_Trading\_Day is Used:** Simply call the procedure with the symbol of the exchange. Ex: exec open\_trading\_day(‘AFE’);

**Parameters:**

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter Name | Parameter Type | Datatype | Description |
| exchange\_symbol | IN | VARCHAR2 | Symbol of the stock exchange |

**Pseudo-code:**

Lookup the stock\_ex\_id using the exchange\_symbol

For each stock that is traded on the stock exchange, insert the stock\_ex\_id, stock\_ids, current date (truncate the time), current date and time, and current price (the price the last time the stock was traded) into STOCK\_HISTORY (stock\_ex\_id, stock\_id, trading\_date, time\_open, open)

#### Close\_Trading\_Day Procedure

**Purpose:** Records the closing price, daily high, daily low and the total volume traded for the day.

**When Close\_Trading\_Day is Used:** Close\_Trading\_Day must be called as close as possible to the time when the exchange closes. It records the closing time, and uses the SYSDATE. If we do not run Close\_Trading\_Day close to the time of the actual closing then we will have to manually update the closing time. It would be best if Close\_Trading\_Day were scheduled to run automatically but it is beyond the scope of this course to use the job scheduler.

**How Close\_Trading\_Day is Used:** Simply call the procedure with the symbol of the exchange. Ex: exec close\_trading\_day(‘AFE’);

**Parameters:**

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter Name | Parameter Type | Datatype | Description |
| exchange\_symbol | IN | VARCHAR2 | Symbol of the stock exchange |

**Pseudo-code:**

Lookup the stock\_ex\_id using the exchange\_symbol;

UPDATE STOCK\_HISTORY

SET high = MAX price for stock for the day,

Low = MIN price for stock for the day,

Close = price for stock’s last trade,

Volume = SUM of shares for stock for the day,

Time\_Close = current date and time

FOR this day and this stock exchange;

# Part 2: First Day of Trading on AFE

We have had our first day of trading!

Trading is simulated by: **TradeOrder\_FirstDayTrading.sql**

Frank Vernon made several orders to both buy and sell but one of his buy orders was very large.

1. Track down all of the trades that were needed to complete that order. Determine the total cost to Frank for that one order, and the average price per share.   
   Although it is challenging, you may be able to write track down all trades in a single query. Look at **Database Code/CTE\_recurs.sql** (ANSI standard) or **Database Code/connectBy.sql** (Oracle specific) for hints on how to write such a query.
2. Write up a “standard operating procedure” that can be followed whenever we need to track down the trades that are generated by a order. If you are able to write a query or a stored procedure then your S.O.P. will be very easy to write.

# Project Submissions

## Part 0

Complete Part 0 and submit your answers to your trainer by the end of day 1. Your trainer will evaluate your answers and then provide feedback as you work on part 1.

## Part 1a

Please submit your answers to questions in an MS Word document by the end of day 3. Because of the nature of the questions, your answers will be evaluated subjectively as per the rubric below. Your descriptions of tables will be marked individually and received up to four points each. Answers to each question will also receive up to four points. Wherever possible, the following RUBRIC will be used:

|  |  |
| --- | --- |
| The answer is correct and complete with a clear, easy to understand explanation using language that is appropriate for business communication | 4 points |
| Answers every element of the question but the answer is not easy to follow or uses language that is not appropriate for a business setting | 3 points |
| Answer partially correct, but misses a minimal amount of elements from the question | 2 points |
| The answer is mostly wrong but has a small part that is right. Either (a) the answer is clearly written and shows you have most of the answer wrong or (b) the answer is not clear and only a small part of your answer can be determined to be correct | 1 point |
| The answer is clearly completely wrong, or the answer is unclear so it cannot be determined whether you have anything right. | 0 points |

## Part 1b

Please submit the DDL, SQL and PL/SQL code that you write in a single .SQL file by the end of day 4. Your code will be evaluated while you work on part 2.

## Part 2

Submit the SQL or PL/SQL code that you used to track down Frank Vernon’s very large buy order as a single .SQL file.

Submit your “standard operating procedure” document as a Word document.