EVE Tools v0.1

Created by Sean Lindsay

Executive Summary

EVE Online is an MMO known for its sheer complexity and steep learning curve, which have earned it the nickname “spreadsheets in space.” It seems only fitting that my project aims to add another spreadsheet to the player’s experience. While the game encompasses makes possible a wide array of options for space-based combat, what actually interest me is the player-driven economy built into the game. As an economics major with a strong interest in business and entrepreneurship, I was primarily drawn to the industrial side of EVE where players buy and sell the in-game resources and items they acquire, and use them to produce items for sale or use. Unlike most business or economic simulators, this market is robust and interesting because it is not the sole purpose of the game. Rather it is the product of natural demand from other players for goods and services that can most easily be acquired from other players, rather than the game itself.

However, the items produced in the game are completely homogenous, meaning that differentiation and branding are impossible. As a result, making a profit in manufacturing is cutthroat and relies on careful analysis of profitability and demand. This spreadsheet was designed to assist in the identification of profitable items to produce at any given time. As prices and conditions shift, the profitability of producing any given item can change dramatically, so being able to generate a profitability report on-demand is extremely useful.

The primary functionality of the current version of this project is the production of profitability reports on any item that can be manufactured. The report includes a wide array of options to allow the user to customize the report, which is particularly important since a large number of these factors can greatly affect the overall profitability. In the interest of usability, however, the report will generate a “default” profitability statement when it is first opened, so no real intervention is required if a quick look is all that is required.

The other major functionality of the project is the management of the extensive combined database that makes the profitability reports possible. This all occurs “under the hood” of the program on hidden sheets, and the average user should rarely have to use these functions aside from the ability to update the web price queries. However, this is where the

bulk of the work for this project has gone, and now that it is done, it should be significantly easier to generate additional reports and features based on this database.

In creating this project, considerable effort was put into making the tool accessible and adaptable. While I had originally hoped to include more functionality than just the profitability report and database functions, I decided to focus on making the project professional, user-friendly, and relatively "bulletproof," rather than include these additional features.

I've spent well over 100 hours on this project, and I've greatly enjoyed the process so far. I plan to continue working on it to add the additional features I would like to see, such as the ability to maintain a "favorites" sheet in the workbook, export generated reports to an excel sheet or other formats, and the generation of quotes and bookkeeping sheets. I also plan to publish this project to the general EVE community at some point, hence the focus on usability. I hope you, the end user, find this project an interesting and useful tool as you play the game.

Implementation

Welcome to

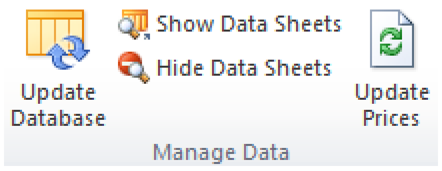
EVE Tools

version 0.1

Created by Sean Lindsay

In discussing the VBA functionality created for this project, it will be most helpful to start with the database management tools, since most of the remainder of the project is based on this groundwork. This part of the project is relatively unglamorous, but it makes it possible for the profitability reports (as well as many other potential reports) to be created with relative ease. The time it took to get this foundational element working probably would not have been worth it if the profitability reports were the only goal, but I intend to expand this project in the future and this database system will make this considerably easier.

Managing the Database



The Manage Data group on the EVE Tools ribbon

Updating the Database

This function, available on the EVE Tools ribbon, allows the user to re-build the entire hidden database maintained in this workbook. The user can select four sheets that have been copied or moved into the workbook, or they can specify a folder which contains each of the four necessary sheets as a separate file.

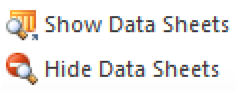
In normal use, the user should rarely need to use this function, but its existence makes the generation of reports possible. This function will consolidate and organize the information from these four sheets (published by designers of the game, CCP) into a single merged sheet, enabling a quick, easy method to determine production requirements. The resulting sheet contains over 1100 columns and 3000 rows, and more importantly it is created in a dynamic process. The single most useful aspect of this new sheet is that it is organized with one item per row, so all production information for an item can be retrieved from a single line.

The reason for creating a dynamic database generator is that the game is periodically updated and expanded, which can change the information contained in this database. A static database, therefore, is only useful until the game is updated, and re-creating the database without this tool would be an incredibly laborious task.

This tool is quite complex because it seeks to avoid dependence on hard-coding. A large number of column headings and other positional information are found using MATCH functions in the creation process, allowing the database to be created even if additional column fields are added to the source sheets by the game designers. Furthermore, other coders can use this functionality to build a consolidated data sheet from which to design their own projects. In fact, the resulting formatted database would be quite useful even to typical users of spreadsheets, since the format lends itself to traditional Excel functions such as VLOOKUP, SUMPRODUCT, etc.

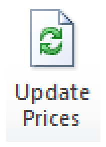
Since this process can take some time, a progress bar was also added so the user is not left wondering if anything is happening. The process can take up to a minute, and will always delete the existing database before creating the new one, so that there is only one database at any given time.

Show/Hide Data Sheets



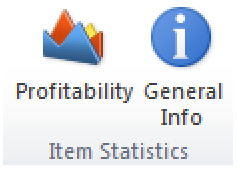
These functions were not originally intended to be included, and were actually added primarily for the benefit of myself and for grading purposes. Since the sheets used by this project are sensitive to changes in these sheets, they have been hidden using the Excel "very hidden" feature. This makes them problematic to get to in the event debugging (or grading!) needs to occur, so I added these functions to make showing and hiding the large number of data sheets easier.

Update Prices



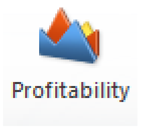
This is the only button in the Manage Data group that I anticipate the end user to use very often. The spreadsheet makes use of two online resources for in-game pricing, EVE-Central.com and EVE-MarketData.com. EVE-Central has some additional detail and information that can be nice in small amounts, but EVE-MarketData allows for a bulk summary of all market prices at once. As a result, the project stores a copy of the EVE-MarketData report at all times for a "quick reference" price, but allows the user to ask for EVE-Central prices when making the profitability reports. EVE-Central requests are created and updated as needed, but the EVE-MarketData query needs to be updated manually, since it is designed to be updated only once or twice a day (rather than every time the user makes a report). In fact, this price updater will check the last time that the prices were updated, and if it has been less than 10 minutes, it will return a polite message box saying that prices cannot be refreshed that often (along with the amount of time before it can be updated again).

Generating Reports



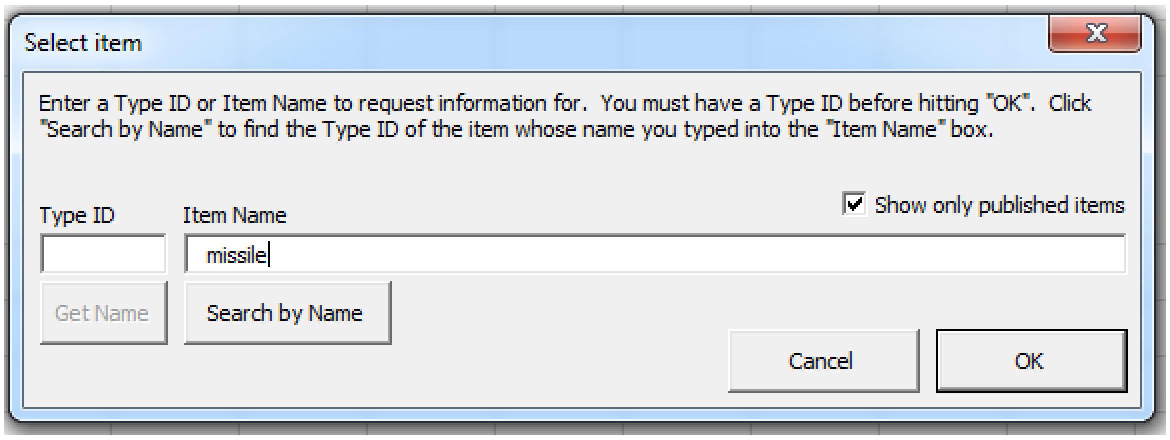
The Item Statistics group on the EVE Tools ribbon

Profitability Report



This is the ultimate goal of this project: a quick and easy way to determine how profitable an item is to manufacture. Clicking on this button will bring up the item selection window, which allows users to pick the item they want a report for. This is somewhat problematic, since the database generally finds items by their Type ID number, so the "Search by Name" button allows users to search for items, and then retrieves the Type ID of their choice for them.

The user also has the option of entering a Type ID, and they can verify that they have selected the right Type ID by clicking "Get Name," which will show that item's name in the Item Name box.



Select item

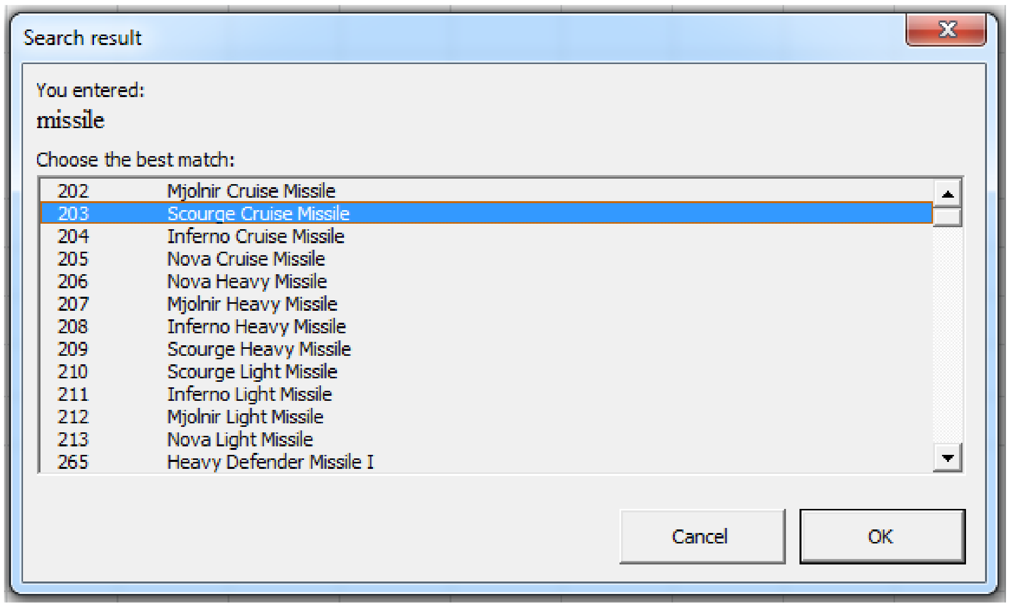
Enter a Type ID or Item Name to request information for. You must have a Type ID before hitting "OK". Click "Search by Name" to find the Type ID of the item whose name you typed into the "Item Name" box.

Type ID:

Item Name: missile

[✓] Show only published items

The Select Item form



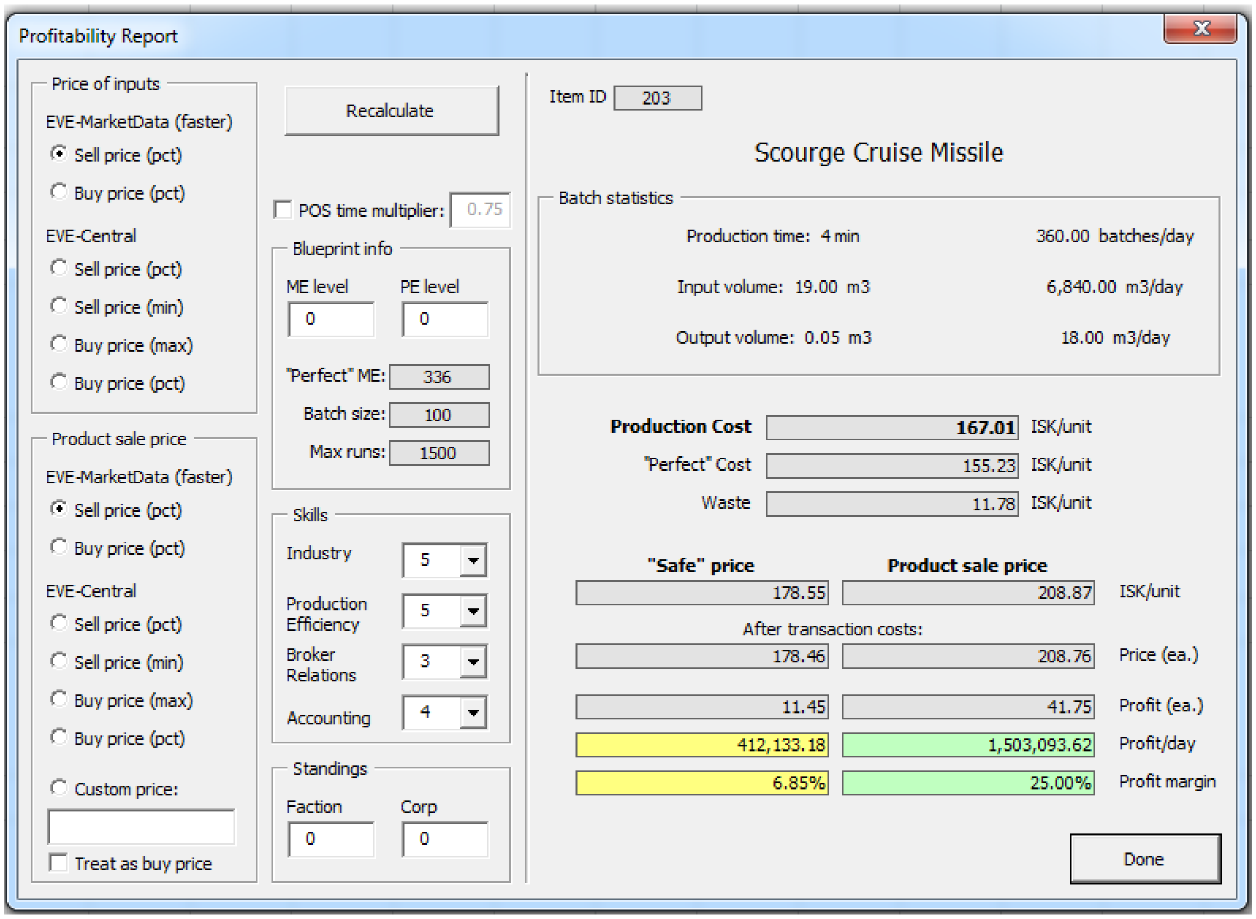
Search result

You entered:
missile

Choose the best match:

202	Mjolnir Cruise Missile
203	Scourge Cruise Missile
204	Inferno Cruise Missile
205	Nova Cruise Missile
206	Nova Heavy Missile
207	Mjolnir Heavy Missile
208	Inferno Heavy Missile
209	Scourge Heavy Missile
210	Scourge Light Missile
211	Inferno Light Missile
212	Mjolnir Light Missile
213	Nova Light Missile
265	Heavy Defender Missile I

Pressing "Search by Name" brings up search results and allows the user to choose one



Profitability Report

Price of inputs
EVE-MarketData (faster)
[✓ Sell price (pct)]
Buy price (pct)
EVE-Central
Sell price (pct)
Sell price (min)
Buy price (max)
Buy price (pct)

Product sale price
EVE-MarketData (faster)
[✓ Sell price (pct)]
Buy price (pct)
EVE-Central
Sell price (pct)
Sell price (min)
Buy price (max)
Buy price (pct)
Custom price:
Treat as buy price

POS time multiplier: 0.75

Blueprint info
ME level: 0
PE level: 0
"Perfect" ME: 336
Batch size: 100
Max runs: 1500

Skills
Industry: 5
Production Efficiency: 5
Broker Relations: 3
Accounting: 4

Standings
Faction: 0
Corp: 0

Item ID 203

Scourge Cruise Missile

Batch statistics
Production time: 4 min — 360.00 batches/day
Input volume: 19.00 m3 — 6,840.00 m3/day
Output volume: 0.05 m3 — 18.00 m3/day

Production Cost **167.01** ISK/unit
"Perfect" Cost 155.23 ISK/unit
Waste 11.78 ISK/unit

	"Safe" price	**Product sale price**	
	178.55	208.87	ISK/unit
After transaction costs:	178.46	208.76	Price (ea.)
	11.45	41.75	Profit (ea.)
	412,133.18	1,503,093.62	Profit/day
	6.85%	25.00%	Profit margin

The all-important Profitability Report

Once an item has been selected, the system will bring up a default Profitability Report for that item, which contains abundant profitability statistics, along with the ability to customize and recalculate the parameters going into the report. While the report may appear overwhelming to those unfamiliar with the EVE manufacturing process, this report condenses the most vital and useful information down to a very readable and accessible form for experienced users.

The left-hand side of the report is dedicated to selecting the parameters used in calculating profitability. The left column allows the user to specify what prices to use, both for the inputs (items going into the manufacturing job) and the output (the end product). These default to the EVE-MarketData options, since this data is kept in the workbook and thus doesn't generate a web query unless the user manually hits the Update Prices button on the ribbon. However, the more detailed EVE-Central statistics can be selected, and pressing "Recalculate" will generate a web query for the appropriate items and populate the new prices to the "Production cost" and "Product sale price" columns in the report. The right column houses the "Recalculate" button, along with a number of parameters which affect both the cost and the time required to build items.

The right-hand side of the report shows important information about the profitability of the item, as well as some helpful batch statistics that help the user understand the logistical side of things. The production cost is given, along with a comparison to "perfect" costs – i.e. what the item would cost to produce under optimal parameters. From this, the waste can be calculated, which helps the user understand if substantial savings could be made by improving the blueprint or skills involved.

For the actual profitability breakdown, two different sale prices are given. The first is a "safe price," which is the buy (percentile) price – the most conservative option available. This is always given as a baseline reference, but the "product sale" price can be adjusted using the settings on the left side of the report. In fact, a custom price can be entered, allowing the user to see if the item is profitable at any price, not just current market prices.

Below these, two important profitability metrics are shown besides simple profit per unit. The first is profit per day, which is the profit that would be made if a days' worth of the item were produced and sold. This is an important metric to consider in maximizing profits, since production times can vary greatly. Also given is a profit margin, which the total margin earned over the base production cost. Some items seem very profitable in a profit/day sense, but return too little relative to the investment to be worthwhile. These last two boxes are colored based on profitability – fields with negative profitability show as red, while fields with modest and good returns show up as yellow and green, respectively.

General Info

This is an unremarkable report generator, which simply brings up a small window with basic information about an item. However, it highlights the potential to quickly and easily generate reports when the database is managed well. The code for this report is quite simple, and it would be possible to generate better and more detailed reports with a minimal time investment in the future, now that the database groundwork has been laid with this project.

Lessons Learned and Difficulties

This project was a massive and complex undertaking, but it has been immensely rewarding. An operating knowledge of VBA (or any programming language, for that matter) requires a lot of hands-on experience, and this project has provided a great objective to work towards while giving me that experience.

First and foremost, I was initially daunted by the prospect of creating user forms, and I generally tried to avoid them wherever possible. However, as I learned more about them and encountered difficulties in formatting sheets to behave like I wanted, I began to realize the true power that user forms give to VBA. Ultimately, user forms became an integral part of my VBA project, and it would have been far more difficult and frustrating to accomplish the same system on a worksheet.

The amount of work that had to go into the hidden database manipulation surprised me. I learned a lot about compartmentalizing procedures, which is incredibly important for a number of reasons. First off, it vastly improves the readability of the code, which is critical while it's in development. Second, it allows oft-repeated functions and lines of code to be reused, so they don't have to be re-typed or hunted down and copied. Third, it makes debugging easier, since it can help isolate the problem better than if the whole code were all in one sub.

Some of the more frustrating and difficult parts of the process were debugging, making the code dynamic, and "bulletproofing" the code, although I got a lot of satisfaction from the latter two. Bugs are inevitable in code, and I pretty much came to expect that the first time I test-ran a sub, it would not work properly for one reason or another. Tracking them down is difficult and time-consuming, but I learned to use a number of online resources (Stack Overflow proved to be particularly helpful) to help out.

Making the code dynamic was much harder than I had previously anticipated, and I ended up using a large amount of variables to pull this off. By “dynamic” I mean that instead of hard-coding column references, I would use identifiers such as column names to “find” the right column before doing something. I hope that this will make the code significantly less vulnerable to database changes and other unforeseen problems.

Bulletproofing is also a tedious and time-consuming process which often serves to clutter up the code quite a bit, but it is very rewarding to be able to run through all the major functions of my code without encountering ugly VBA errors. In addition to bulletproofing, I spent a lot of time learning how to make things user-friendly. It’s easy to write code that *I* know how to use (i.e. “okay, I just need to put this number in A5 and it will give me what I want”), but much less easy to make it accessible to the average user who has no idea about how it all works. Ultimately, that meant making good use of userforms. I do plan to distribute this project to the general EVE community once I’ve had a chance to add some more functionality, so I’m happy with the level of usability I’ve been able to achieve.

Finally, I learned that sometimes you just can’t do everything you want to in the time you have. I would have liked to include a number of additional reports, but there quite simply wasn’t time. Coding in VBA can be a time-consuming process, especially when you’re just learning how to do things. I chose to focus on quality rather than quantity of features, so I am satisfied with the final product. It maintains the core functionality that I wanted, despite lacking some of the additional features I would have liked.

Assistance and Thanks

Virtually all of the code in this project is mine and mine alone. The only code I explicitly borrowed from another source was a function written by Peter Albert on stackoverflow.com, which allowed me to remove the [X] to close button in the top right corner of my progress bar (exiting the progress form would abort the database compilation, which I didn’t want). I used stackoverflow.com and a number of other online resources such as the Microsoft web documentation for VBA extensively, both for troubleshooting and for help understanding various concepts. I also used http://wiki.eve-id.net/Equations for help with the complex production formulas EVE uses. I would finally like to thank Nick Jones for his occasional thoughts, and of course Dr. Gove for this teaching this incredibly useful class.