Hi and welcome to my page where I showcase some of my home projects, skills and use of technology that I hope you’ll find interesting and desirable for your project or organization!

A little bit about me ☺

With over two decades of experience in software development and data-oriented systems, I specialize in data, technical migration, ETL processes, and database engineering.

My professional journey has focused on empowering organizations with reliable, efficient, and well-optimized data solutions. I’m industry agnostic, having plied my trade in Finance, insurance, marketing, retail, and even defence. Previous engagements include contributions at Barclays, AIG, BAe Systems, John Lewis, where I brought an emphasis on quality, enhanced system performance and streamlined cross-team communication.

A recent role involved providing technical and functional consultation at Lenvi, where I facilitated the migration of lending service providers to an Azure-hosted loan administration platform. I designed data transformation systems, developed documentation, and conducted knowledge transfer sessions while leveraging my expertise in Oracle databases and programming.

Having completely numerous professional courses and holding Oracle OCA certification, I am committed to delivering excellence in data migration, database management, and technical consultation.

Describe the case study overarching aim:

As a hypothetical and purely academic problem to solve, I thought I’d try to use AI to win the Euromillions lottery ☺ Best case outcome is obvious, with the worst being that you get to see how I went about trying to solve the problem here. Win, win!

Oh and to top it off, I thought I’d give myself the constraint of everything has to be free. No licences!

Description of physical architecture:

For fun, and to make the problem more interesting I decided to build this lottery number predictive system over multiple platforms. In the cloud, on my local machine and using docker containers giving a complex heterogenous hybrid system to reflect what is typically out there in real enterprise environments. Completely unnecessary, but cool to play with.

My development machine is a Silicone based mac doubling as substitute for on prem servers

There are 3 RDBMSs used in this system for data excluding the airflow metadata db;

* Oracle Cloud free tier hosted Oracle 23ai database
* A locally hosted docker image of an sql server database
* A cloud hosted postgres db v17.6. Hosted for free on aiven

Running a conda distribution of python v3.10.9 for the application code with Java version 17

I attempted running airflow version 3.1.0 in the localhost over a second conda environment Unfortunately, this had many issues in an ARM environment and is unstable.

In the end I opted to use a docker compose image of airflow 2.9.3 as baseline for the airflow environment as it is documented as stable for ARM hardware. It was then extended via dockerfile and yaml files to give all the additional python modules and java runtime required by the system.

I used Visual Studio Code (VS code), with several extensions to cater for the multiple db types. However, I dropped it down to just the one for SQL Server as the extensions conflicted. The final dev setup used SQL Developer for Oracle, and TablePlus for Postgres

All the code was configured using github.

Scenario 1 – Getting the draw data.

I obtained the historic draw numbers from the National Lottery website, but they only provide draw data in consumable form (csv) going back a limited period (about 6 months of draw data). This initial set of files from the national lottery website can be considered the highest quality of data, while the quantity is small. As such, I had to find another source if I wanted better/ comprehensive coverage of all draws. In the interim I created a target table in postgres for this data and code to load it.

Code;

create\_EuroMillions\_lottery\_table\_PG.psql

load\_source1\_Euromillions\_lottery\_data\_to\_postgres.py

I obtained a csv data source from a public google drive with greater coverage, but visual inspection of the data with some adhoc sql queries, showed some inconsistencies with the National Lottery website data. Thus, this source could not be trusted and was discarded in favour of finding another alternative source.

Code;

create\_EuroMillions\_lottery\_table\_PG\_2.psql

load\_source2\_Euromillions\_lottery\_data\_to\_postgres.py

Scenario 2 – Getting and validating a comprehensive dataset.

I found the website <https://www.beatlottery.co.uk/> that has full history for the Euromillions lottery. You can view any draw numbers from when the lottery started. Unfortunately, to get a consumable format for the data they charge a fee.

As paying a fee breaks my self-imposed constraint for this project, I decided to create a web scraping application to harvest the data instead. Given the previous problem I would then need to validate its quality by comparing it to the reliable subset of data obtained from the National Lottery website.

The data is scraped and stored in csv. The csv is then loaded in the Oracle instance and finally the scraped data in Oracle is validated against the National Lottery website data that is stored in Postgres

Code;

scrape\_euromillions\_data.py

load\_euromillions\_data.py

validate\_data.py

Scenario 3 – Analysis and prediction

At this point it’s time to start looking at the data and trying to make some predictions.

For the human readable analysis, I began by fetch the comprehensive validated lottery data from the Oracle db into a pandas dataframe to, clean, and perform a deep statistical analysis. It answers questions like:

* Which numbers have been drawn the most or least often overall?
* Did the frequency of numbers change from one year to the next?
* Are certain numbers currently on a "hot streak"?

The results are then plotted using matplotlib as easy-to-understand charts and a detailed log file, providing insights into the historical performance of EuroMillions numbers.

Code;

analyze\_euromillions\_data.py

For the first predictive method I created a statistics-based tool, again using pandas, that analyses historical lottery data to generate number suggestions based on different logical patterns. It interactively tells you which numbers are historically "hot" (frequent), "cold" (infrequent), or "overdue," and then presents a ticket based on your preferred strategy.

Code;

predict\_euromillions\_numbers.py

The second predictive model builds a prediction system using the sklearn module. It creates a rich set of historical features for every past draw and uses them to train dozens of individual machine learning models. Each model becomes an "expert" on a single number, learning the patterns that tend to precede that number being drawn. The final prediction is a "committee vote" where the numbers with the most confident expert models are chosen.

The user can interactively choose if they want to train the model using all the data or just the Tuesday, or Friday draws.

Code;

predict\_ml\_euromillions.py

The final predictive model builds a more powerful prediction engine using a neural network. It processes historical data to create a rich set of features, then trains two specialized "brains"—one for main balls and one for lucky stars—to learn the complex relationships between past draws and future outcomes. The final prediction is based on which numbers the trained models "believe" have the highest chance of being drawn next.

Code;

predict\_pytorch\_euromillions.py

Scenario 4 – Moving data between db systems with intermediary files

Purely for academic purposes as there is no practical purpose in the context of this project.

It connects to an Oracle database, makes a complete copy of the EuroMillions draw history, and saves that copy locally as both a CSV and a JSON file. This prepares the data to be moved or analysed by other systems without needing a constant connection to the original database.

I use PySpark to efficiently move data from a JSON file into a temporary staging area in SQL Server. Then, use a sophisticated MERGE command to intelligently update the main database table, ensuring that the data is always up-to-date without creating duplicates.

Code;

export\_oracle\_to\_local\_files.py

load\_json\_to\_sqlserver.py

Scenario 5 – Automation

At this point, everything had been developed to run locally in my “dev” environment. To more closely mimic a real world production environment, I wanted to automate some of the potential workflows using airflow.

As I am using a docker compose airflow environment, I extended the base image using a dockerfile and requirements file with the required components e.g. Java runtime, and python modules.

There were a number of code changes required to permit the code to run both locally and via airflow so I opted to have single script component dags initially for unit test and then created more workflow driven dags.

The workflows are;

A complete "Extract, Load, and Validate" process.

It reliably fetches new EuroMillions results after each draw, loads the primary data into an Oracle database, loads supplementary data into a PostgreSQL database, and then runs a final validation step to confirm that all systems are synchronized and the data is trustworthy.

Code;

euromillions\_ingestion\_dag.py

load\_source1\_Euromillions\_lottery\_data\_to\_postgres.py

scrape\_euromillions\_data.py

load\_euromillions\_data.py

validate\_data.py

Analysis and prediction process.

This provides a convenient way to run all the project's analysis and prediction tools at once. It's a manually triggered pipeline that first performs a general data analysis and then runs three different prediction models in parallel, allowing the user to easily configure the behaviour of each model through the Airflow interface.

Code;

euromillions\_analysis\_dag.py

analyze\_euromillions\_data.py

predict\_euromillions\_numbers.py

predict\_ml\_euromillions.py

predict\_pytorch\_euromillions.py

Extract, Transform, Load (ETL) from Oracle to SQL Server.

It extracts data from Oracle, temporarily saves it to a file (minimal transformation), and then loads it into SQL Server. This could be used to ensure the SQL Server database is kept in sync with the master Oracle database on a daily basis for example.

euromillions\_export\_dag.py

export\_oracle\_to\_local\_files.py

load\_json\_to\_sqlserver.py

Potential further enhancements:

* Adding constraints to the db tables PK, and Unique constraints on date, draw number
* A better way to handle the delta than getting all the results for comparison
* Adding a schedule to the dags for complete automation after each draw
* Refactoring all the code for there to be shared modules with database connections and other functionality, perhaps even utilities
* Additional database tables, processes to track predictions and their accuracy
* Make a Kubernetes cluster to automate running the entire system including startup, shutdown etc

As this webpage is still active you can guess I haven’t been successful in predicting the winning numbers, but hopefully I’ve accomplished my other goal of showing off my skills and how I’d approach creating solutions to problems.