**Analysis of Job Popularity as a Software Developer in Canada**

November 29th, 2023

## **Abstract**

In this project, two job board websites, Workopolis, and Job Bank, were web scraped using Selenium Package in Python running on the WebDriver for the Google Chrome browser to gather insights about software developer jobs in Canada. Data is gathered from these 2 websites weekly. An aggregate integration of weekly data was created after processing the data. This aggregate data was then stored in the Microsoft SQL Server database. Lastly, an analysis and visualization of the data was completed.

**Table of Contents**

[**Abstract** 2](#_Toc177661940)

[**Introduction** 4](#_Toc177661941)

[**Data Research and Integration** 4](#_Toc177661942)

[**Data Collection** 5](#_Toc177661943)

[**Data Storage and Maintenance** 5](#_Toc177661944)

[**Data Quality** 6](#_Toc177661945)

[**Data Analysis and Visualization** 6](#_Toc177661946)

[**Scale-up Estimations** 7](#_Toc177661947)

[**References** 13](#_Toc177661948)

## **Introduction**

The focus of this project will be to perform an analysis of jobs within the IT field in Canada (specifically developer jobs) that have been posted recently to identify what jobs have grown in popularity as time passes. The data is on jobs in Canada, like job title, company name, location, and approximated salaries. The data is being retrieved through web scraping techniques through the Selenium package in Python by using WebDriver for the Google Chrome browser. The data collection process is being automated using Python scripts: Two Python scripts were created to scrape the data of each website and then saved into two CVSs. After data collection, two Python scripts were created to clean each dataset in an automated manner. Once cleaned, these datasets are merged into a single weekly data set. Before populating the Microsoft SQL Server database with this data, another integration of weekly data is done. Lastly, it was decided to use Tableau for the visualization and analysis stage.

## **Data Research and Integration**

The initial idea was to use Indeed as one of the data sources for this project. However, the structure of their web code was dynamic and complex, so extracting data from Indeed was a little complex. Therefore, alternative websites were sought to collect the primary data. Furthermore, an attempt was made to extract the data via an API tool, but the API used did not work for the chosen websites, and it was decided to use web scraping as the alternative technique.

There are two levels of data integration in this project. First, it integrates data from two websites, and second, it integrates weekly data into an aggregate version.

The employment job posting websites Workopolis and Job Bank were chosen for extracting the data through web scraping. After collecting this data, it was saved into two separate CSV files for each website. Both datasets are cleaned and combined into a single CSV file using a Python script. The mergeWeeklyData.py Python script was used to integrate data from two sites into a single CSV. However, as the data collection was done weekly, the integration of weekly data (job postings from two websites) into an aggregate compilation of job data was done using MergeJobData.py, another Python script. This aggregate version has the integrated data from all weeks throughout the project.

For the integration of weekly data into an aggregate version, the issue was that job postings from previous weeks could still be posted on the websites. So, a direct approach to integrating the data would result in duplicated jobs in the aggregate version, leading to a flawed analysis. To solve this, the columns job title, company, and location were used to identify unique job postings and eliminate duplicates, resulting in the final version used for analysis.

## **Data Collection**

An average of 250 rows of data are being collected each week combined from both sites. Data is being collected using the Selenium package in Python through the WebDriver for the Google Chrome Browser. This process was automated using two Python scripts as data is collected from two data sources(websites). During the project, 8 weeks of data were collected, with columns such as Job title, Company, Location, and estimated salary. The JobsToCSV1.py and JobsToCSV2.py were used to scrape job data from Workopolis and Job Bank, respectively.

**Technical details:**

The Chrome web driver should be downloaded from <https://chromedriver.chromium.org/downloads>.

The Chrome web driver should be downloaded for the matching version of the local Chrome web browser. Also, during data collection, as selenium is being used to navigate through pages in an automated manner if the speed of the internet is slow, the web driver wait time in data collection scripts JobsToCSV1.py and JobsToCSV2.py should be increased so the pages are given time to load. The next page button is available for navigating to web pages. If the internet speed is slow and the wait time is not increased, the Python script will stop due to not locating the next page button. This can be observed as the number of pages web scraped are displayed after execution of the Python script. So, checking if the number of web scraped pages matches the website's pages is recommended.

## **Data Storage and Maintenance**

The data is being stored in Microsoft SQL Server. Microsoft SQL server was mainly chosen for storage as all team members are familiar with this technology. Also, the requirements of the project, as in the operations being done in this project, should be compatible with SQL server and due to this, rather than using a database that members of the team are not familiar with and might have difficulty in adopting a new tool we choose to use SQL server.

After all data cleaning and integration of weekly data into an aggregate version, the MS SQL server has a feature to import a CSV file to create a table. As the data was cleaned and integrated using Python scripts, this data can be imported. No additional cleaning was required into the MS SQL server database using the import flat file feature in the task option, which can be found by right-clicking on the database.

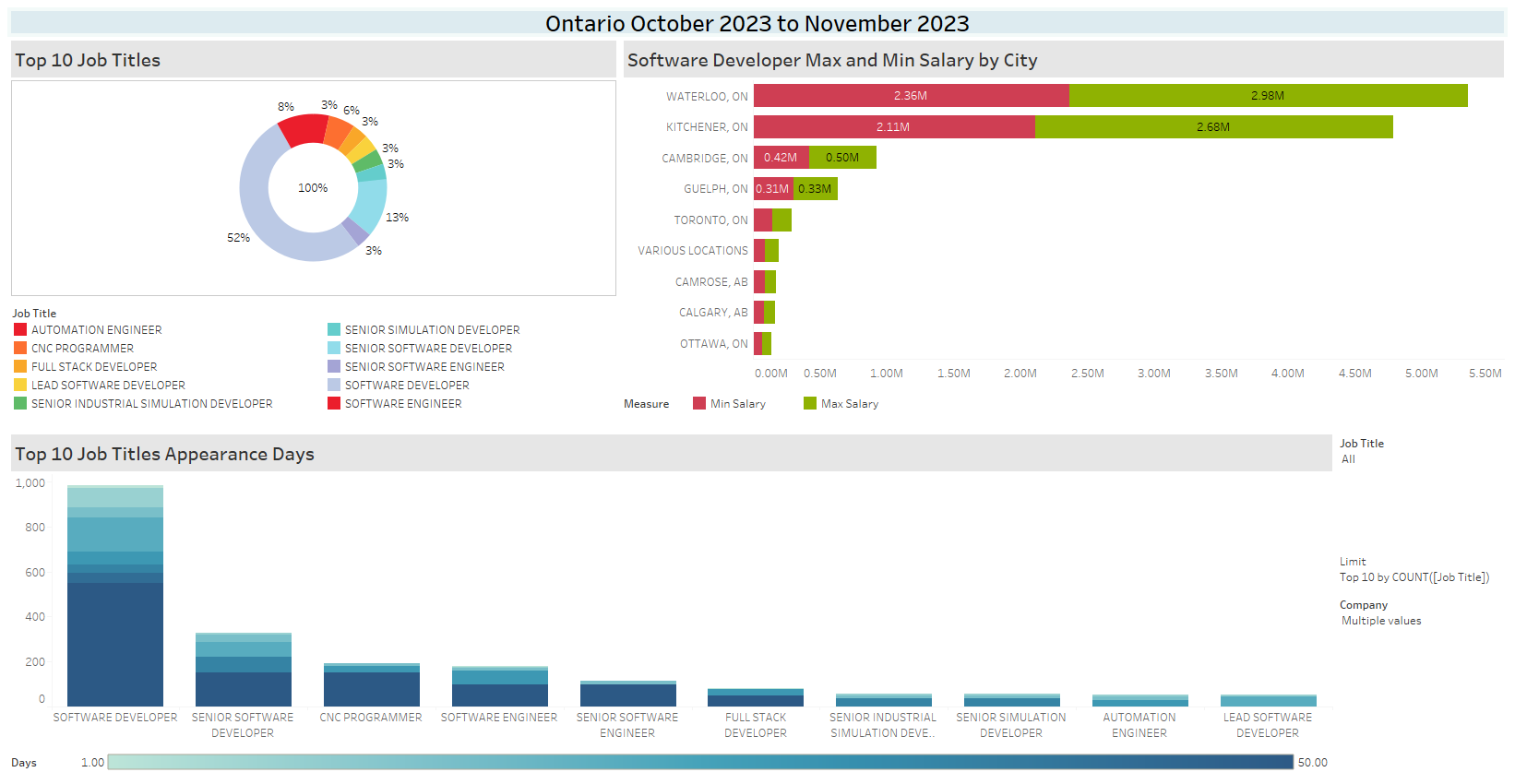
## **Data Quality**

To maintain the quality of the data, cleaning of all the information contained in the two datasets created from web scraping is being performed so that the population of the database is easier. To carry out this process in an automated and more efficient manner, Two Python scripts, clean\_jobbank.py and clean\_workopolis.py, were created, one for cleaning data web scraped from Job Bank and another for cleaning data scraped from Workopolis. While cleaning the data, a Retrieved Data column was created that stored the date the job posting was retrieved from the site. Only after validating that the data has been cleaned properly are they merged into weekly job data using another Python script, mergeWeeklyData.py. As data cleaning is an iterative process, more cleaning was required in the later stages of the project, and some data cleansing steps were done while integrating the data. During the final integration of weekly data into an aggregate version, another column, the last appearance column, was created. This column stores the max retrieved date for each unique job posting. This was done roundabout because if the number of days since the job was posted would differ as data is collected weekly, it would be inaccurate as data collection was done weekly and could have duplicate job postings.

## **Data Analysis and Visualization**

It was decided to use Tableau as a tool to visualize and analyze our data to take advantage of the knowledge of all the members of the group on the software.

The visualizations were created using the Excel file “FinalJobData.xlsx” which contains data from October 4th to November 22nd.



Three different worksheets were created to show the insights that were obtained with the data collected:

1. **Top 10 Job Titles.** Shows the 10 most popular job titles that were posted.
2. **Software Developer Max and Min Salary by City.** Shows the maximum and minimum salary by city for the position of Software Developer
3. **Top 10 Job Titles Appearance Days.** Shows the total number of days each of the top 10 jobs appeared during the data collection period.

A dashboard was created to show the worksheets mentioned above together so that it is easier to visualize these insights. In this dashboard, it can be clearly seen that the leading role is Software Developer, which is included in 45 rows of the dataset, representing 52% of the top 10 job titles. Furthermore, it is observed that the second most popular role is Senior Software Developer.

On the other hand, Waterloo and Kitchener have the best maximum and minimum salaries compared to the rest of the Ontario cities in the dataset.

Finally, it is shown that the Software Developer role remained posted from 1 day to 50 days by different companies and for different cities in Ontario.

## **Scale-up Estimations**

The weekly data is 22kb of data and 8kb of index data, which is a total of 30kb of data per week. An SQL server can handle up to 524 petabytes of data. But hypothetically, the database does grow big enough. Compression and indexing can be used to make queries faster. It could also be considered switching to a NoSQL database but as the schema does not change and the format of data that is being stored is compatible with the SQL server, there is not a compelling reason to switch to a different database.

**Scale up**

To increase the scale of this project we can add more websites that we will scrape data from.

Furthermore, we can break down some columns to give us more details about each posted job. In addition, we can create different tables for different types of jobs and not stay restricted to software developer jobs. Since our datasets do not take up a lot of space, we can continue to store them on our local disks. When that starts to become a problem, we can store our data in an external hard drive. That would be a cheap option for us to store our data. Eventually, when storing it in an external hard drive poses a problem, we can start thinking about which cloud service we can use to store our data. As long as we are fine with another organization holding our data.

**Scaled up Storage**

**Storage Cost Estimation for Google Cloud Platform (GCP)**

Storage Cost Details:

Storage Class: Standard Storage

Cost per GB per Month: $0.023

**1. Storage Cost Details:**

• Storage Class: Standard Storage

• Cost per GB per Month: $0.023

**2. Daily Storage Needs:** Given daily storage needs: 30 KB/week x 1 week=4.2857 KB/day

3. Intermediate Storage:20% of daily storage needs: 0.2 x 4.2857 KB/day = 0.8571KB/day

4**. Monthly Storage Cost Calculation:**

• Using the formula: Monthly Storage Cost = Daily Storage Needs × Cost per GB × 30

• Substitute the values: Monthly Storage Cost = 4.2857 KB/day × (1024KB) $0.023/GB/month × 30 days

• Calculating the result: Monthly Storage Cost $0.00084 per month

**5. Yearly Storage Cost:**

• Yearly Storage Cost = Monthly Storage Cost × 12 months

• Calculating the result:

Yearly Storage Cost = $0.01008 per year × 5 years

**Yearly Storage Cost (5 Years) = $0.0504 for 5 years**

**Storage Cost Estimation Report with Replication Factor:**

The monthly storage cost with replication can be calculated using the formula:

Monthly Storage Cost=Daily Storage Needs (with Replication) × Cost per GB×30

Substituting the values, we get:

Monthly Storage Cost=8.5714 KB/day× (1 GB/ 1024 KB) × $0.023/GB/month×30 days

**Calculating the result, the monthly storage cost is approximately $0.00168 per month.**

The yearly storage cost with replication is then:

Yearly Storage Cost (With Replication) =

Monthly Storage Cost (With Replication) ×12 months.

Substituting the monthly cost, we get:

Yearly Storage Cost (With Replication) =$0.02016 per year

**For 5 years, the yearly storage cost with replication is approximately $0.1008 for 5 years.**

**Storage Cost Estimation Report for Amazon S3 Intelligent-Tiering - Without Replication:**

Storage Cost Details:

Storage Class: S3 Intelligent-Tiering

Standard Intelligent-Tiering Cost: $0.023 per GB per month

Archive Access Tier Cost: $0.00405 per GB per month

**Daily Storage Needs:**

Given daily storage needs: 25KB/day.

Intermediate Storage: 20% of daily storage needs: 0.2×25 KB/day=5 KB/day.

**Monthly Storage Cost Calculation (Standard Intelligent-Tiering):**

Using the formula:

Monthly Storage Cost (Standard)=Daily Storage Needs × Standard Intelligent-Tiering Cost×30

Substitute the values:

Monthly Storage Cost (Standard)=25 KB/day × (1 GB/1024 KB) × $0.023/GB/month×30 days

Calculating the result:

Monthly Storage Cost (**Standard**)=$0.00197per month

**Monthly Storage Cost Calculation (Archive Access Tier):**

Using the formula:

Monthly Storage Cost (Archive)=Intermediate Storage × Archive Access Tier Cost×30

Substitute the values:

Monthly Storage Cost (Archive)=5 KB/day × (1GB/1024 KB) × $0.00405/GB/month×30 days

Calculating the result:

Monthly Storage Cost (**Archive**)=$0.00077 per month.

**Yearly Storage Cost (Without Replication):**

Yearly Storage Cost (Standard)=Monthly Storage Cost (Standard)×12months

Yearly Storage Cost (Archive)=Monthly Storage Cost (Archive)×12 months

Calculating the results:

Yearly Storage Cost (**Standard**)=$0.02364per year

Yearly Storage Cost (**Archive**)=$0.00924per year

**Total 5-Year Storage Cost (Without Replication):**

Total 5-Year Storage Cost (**Standard**)=Yearly Storage Cost (Standard)×5 years

Total 5-Year Storage Cost (**Archive**)=Yearly Storage Cost (Archive)×5 years

**Calculating the results:**

Total 5-Year Storage Cost (**Standard**)=$0.1182 for 5 years

Total 5-Year Storage Cost (**Archive**)=$0.0462 for 5 years

**Storage Cost Estimation Report for Amazon S3 Intelligent-Tiering - With Replication:**

Storage Cost Details:

Storage Class: S3 Intelligent-Tiering

Standard Intelligent-Tiering Cost: $0.023 per GB per month

Archive Access Tier Cost: $0.00405 per GB per month

**Replication Factor: 2**

**Daily Storage Needs:**

Given daily storage needs: 25 KB/day.

Intermediate Storage:20% of daily storage needs: 0.2×25 KB/day=5 KB/day

Replicated Storage:

Storage needs with replication: 25 KB/day×2=50 KB/day.

Monthly Storage Cost Calculation (Standard Intelligent-Tiering):

Using the formula:

Monthly Storage Cost (Standard)=Daily Storage Needs (with Replication) × Standard Intelligent-Tiering Cost30

Substitute the values:

Monthly Storage Cost (Standard)=50 KB/day × (1GB/1024KB) × $0.023/GB/month×30 days

Calculating the result:

Monthly Storage Cost (**Standard**)=$0.00394 per month

**Monthly Storage Cost Calculation (Archive Access Tier):**

Using the formula:

Monthly Storage Cost (Archive)=Intermediate Storage (with Replication) × Archive Access Tier Cost×30

Substitute the values:

Monthly Storage Cost (Archive)=5 KB/day×2× (1 GB/1024 KB) × $0.00405/GB/month×30 days

Calculating the result:

Monthly Storage Cost (**Archive**)=$0.00155per month

**Yearly Storage Cost (With Replication):**

Yearly Storage Cost (Standard)=Monthly Storage Cost (**Standard**)×12months

Yearly Storage Cost (Archive)=Monthly Storage Cost (**Archive**)×12 months

Calculating the results:

Yearly Storage Cost (**Standard**)=$0.04728per year

Yearly Storage Cost (**Archive**)=$0.01866per year

**Total 5-Year Storage Cost (With Replication):**

Total 5-Year Storage Cost (Standard)=Yearly Storage Cost (**Standard**)×5 years

Total 5-Year Storage Cost (Archive)=Yearly Storage Cost (**Archive**)×5 years

**Calculating the results:**

Total 5-Year Storage Cost (**Standard**)=$0.2364 for 5 years

Total 5-Year Storage Cost (**Archive**)=$0.0933 for 5 years

**Conclusion:**

Google Cloud Platform (GCP) vs. Amazon S3:

Over a 5-year timeframe, GCP is more cost-effective than Amazon S3 for both replication scenarios.

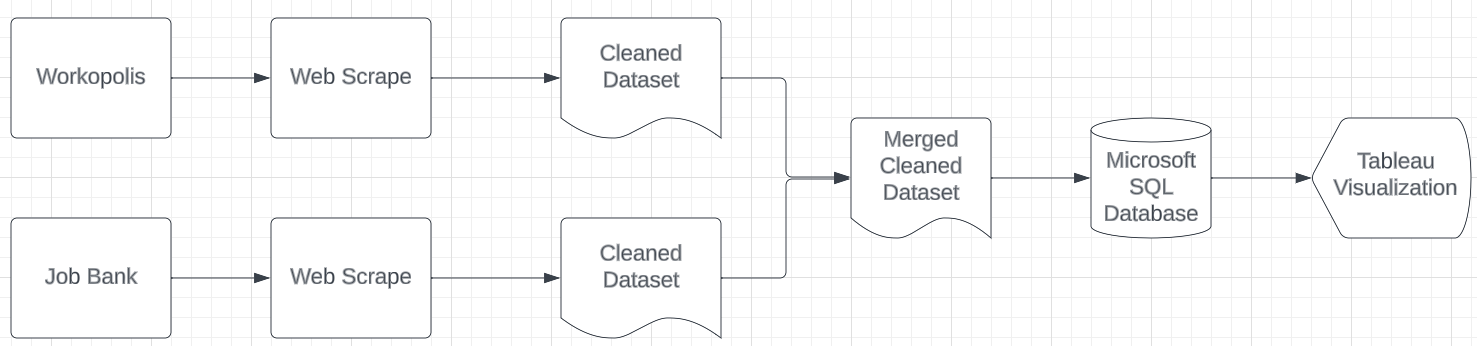
GCP offers much lower costs than Amazon S3 Intelligent Tiering without replication. With replication, GCP still has a cost advantage over Amazon S3, although it is less apparent.

**Impact of Replication:** Replication significantly raises the expenses of both platforms. While redundancy is necessary, it substantially increases total costs.

**Tiering Techniques:** Amazon S3's Intelligent Tiering feature optimizes cost depending on data access patterns. The Archive tier offers lower-cost storage for material that is accessed less frequently.

**Long-Term Prices:** GCP consistently has lower prices across both scenarios and replication factors, making it a more cost-effective alternative for long-term storage needs.

**Diagram**



## **References**

Amazon S3 Simple Storage Service Pricing - Amazon Web Services. (2023). <https://aws.amazon.com/s3/pricing>

Chromium and WebDriver teams. (2023). *Chromedriver*. Chromedriver - WebDriver for Chrome. <https://chromedriver.chromium.org/home>

Dhanashree. (2023, March 20). Web Scraping with Selenium: The complete guide. Nanonets.com. <https://nanonets.com/blog/web-scraping-with-selenium/>

pandas. (2023). *Data Frame*. <https://pandas.pydata.org/docs/reference/frame.html>

Pricing | Cloud Storage | Google Cloud. (2023). <https://cloud.google.com/storage/pricing#regions>

Tableau. (2023). *Tableau Desktop and Web Authoring Help*. <https://help.tableau.com/current/pro/desktop/en-us/default.htm>

Workopolis. (n.d.) All software developer jobs in Elmira Ontario. Web scraped data source. <https://www.workopolis.com/jobsearch/find-jobs?ak=software%20developer&l=elmira%20ontario&job=P5vrTLwm1A1yIaghfCvXVPgnIA-B90B11E-Y3Kfdefn06dPqBQSL3k1H6TrnPBYV>

JobBank. (n.d.) All software developer jobs in Elmira Ontario. Web scraped data source. <https://www.jobbank.gc.ca/jobsearch/jobsearch?searchstring=software+developer&locationstring=Elmira%2C+ON>

YouTube. (2022). *How to import csv file into SQL server*. *YouTube*. <https://www.youtube.com/watch?v=VdvrGgXGRfM&t=8s>.

Zenrows. (2023, October 14). Web Scraping with Selenium and Python in 2023. Zenrows.com. <https://www.zenrows.com/blog/selenium-python-web-scraping#headless-chrome>

‌