Format from -> https://www.youtube.com/watch?v=DWjIU55COSk
Front page
    Name
    Academic year
    Name of guide
    Collage name
Certificate of completion
Certificate by guide or teacher
Declaration by student
    The work done by be is genuine
Acknowledgement
Index Page
    Key Highlightsw
Contents Page
    Each chapter divided into details
List of tables or graphs
Executive summary – Trailer of the entire project
    Total report 30-40 pages -> summary -> max 2 pages
Chapters
    Chapter 1 – Introduction
    Chapter 2 – Company Profile/Literature Review
    Chapter 3 – Research methodology
    Chapter 4 – Data analysis and interpretation
    Chapter 5 – Findings
    Chapter 6 – Suggestions
    Chapter 7 – Limitations(max 4-5)
    Chapter 8 – Conclusion
References
Appendix or Glossary

Internship

Week 1:
Goal – An amazon product listing scrapper to get the rank of product according to the pincodes/regions they lie in.

Projects made –
- A semi working rank finder for products using selenium and another one without any browser automation
- A buybox checker to find out the seller for prodcuts in the current/given pincode/region
- Google sheets App Script to read and write to google sheets to collect and publish data

Steps taken –

- Rebuild old scrapper selenium from scratch
- Optimized ASIN-keyword mapping to reduce the number of searches
- Used rotating proxies to hide IP from the website
- Used reverse engineering to figure out the API endpoints to get to the correct cookies to complete the same task without the use of browser optimization
- Speed up existing scrapper by structuring quires such that fewer of them are required
- Error recovery made so that in case of failure you can continue from where you left or crashed
- Continuous retries to a certain extent in case of failure to try to not crash
- CAPTCHA buypass/completion in case of encounter
- BuyBox scrapper to find out the seller, selling price, ratings, limited time deal, other sellers for the same product, etc from the product pages
- Used non browser automation script to speed up the state management and efficiency of scrapper

Week 2:

Problems faced –

- Missing fields not scrapped/missed by the scrapper
- Amazon CAPTCHA and bot detection
- Scrapper to continue from where it left out
- Ability to limit work done per run
- Need a fresh session for each keyword for ranking scrapper

Steps taken –

- Used parsel instead of BeautifulSoup4 to parse the html for powerful searching using xpath, css and regex
- Added a TOR based proxy to get proxies for the scrapper
- Added a check for empty/bogus pages given by Amazon to limit traffic
- Changed implementation to httpx from requests to use http v2 instead of v1.1 which is used on requests
- Switched to using py files instead of python notebooks for modular approach
- Created global settings to change the settings for different scrappers
- Implemented a reusing approach for the ranking scrapper to try to reduce the number of requests sent to amazon in attempt to increase the scope of scrapping before detection
- Tried implementing an undetected chrome driver to get the initial session cookies for the scrapper
- Created different list of work for each module of the scrapper so that they can lag behind one another and can still start off from where they left off
- Used non browser automation script to speed up the state management and efficiency of scrapper

- Created a backup model where the previously done work is backuped up before new one is started, so that in case of failure or crash we can revert to the backup data
- Found Amazon endpoint to turn off searching history in sessions to reduce the amount of biasing to the keyword ranking

Week 3:

Problems faced –
- No authentication for reading and writing to the google sheets
- No way to see the ranking and delivery time in a meaningful way
- No full India coverage
- Better way to filter data
- Bad coloring on map using looker studio
- Comparative timeline to competitors
- Error prone scrapping
- Too much data for google sheets
- No data to compare our stats too
- Remove most of the dependency from Google Sheets

Steps taken –
- Using google OAuth to sign a user in and use their token for authentication
- Remove google app script for the sheets and instead using Google's API to read and write to the sheets
- Restructured the project to use overlapping sections in one place instead of writing them in multiple places
- Taking data from other sheets to map the pincodes to their corresponding cities and states
- Using googles looker studio and the scrapped data, plotted the data on a map showing
    - Area with good delivery times and worse delivery times
    - Ranking of products according to various states
- Introduced pincodes for all states in india with big states having more than one
- Introduced 2 new filters
    - Seller – Check which seller is selling in which states and what is their time to deliver
    - Time to Deliver – Set a upper and lower limit of the time to deliver to see states with custom ranges
- Included bogus values to set the min and max values for each ASIN making sure that the color gradient is consistent across all the ASINS
- Scrapping the delivery timeline of the top 5 organic ranking products when scrapping for rank
- Quitting scrapping if faces error 3 times
- Google sheets limit for 10mil cells so saving data into csv files
- Getting top 5 items delivery times to compare our delivery time too
- Parse the dates manually and calculate the Time to Delivery in the scripts itself

- Export directly to csv files

Week 4:

Problems faced –
- Very messy project
- Continuous blocking of requests
- Unclear progress
- Distributive package
- Intermediate files written in unclear places
- Getting the status of orders like there delivery status etc.
- Very labor intensive to check for returned orders using order-ids
- Must get keyword search volume from external tools like Helium

Steps taken –
- Proper documentation of the project
- Changing from httpx to curl_cffi that can impersonate different browsers
- Created a progress bar for each of the sessions created
- Slow down the scrapper and changed to single thread mode to reduce outgoing requests, this reducing the chances to get blocked
- Creating a Exe and adding options to share with others
- Get the location of exe during execution and read/write files relative to that path only
- Expanded launched to include order status and items detail using the amazon SP-API
- Getting item return reports to include in the order status script to check for returns on those orders
- Fetching reports for asins, which include the keyword search volume to get the search volume for them

Week 6:

Problems faced –
- File paths not working with a binary executable
- Search volume logic
- Buybox other sellers' price comparison
- Make google sheets entries easily queryable

Steps taken –
- Introduced finding a base path of the binary executable so that all files are created relative to the location of the executable
- Added logic for calculation of search volume click through rate of each product by taking into factor the total percentage contributed by the pincode and the rank of the product for said keyword

- Created a separate google sheet which gets written to comparing all the sellers of the products, there prices, and delivery methods(MFN/FBA)
- Added ability to enter data into google sheets by overwriting old data and not shifting it
- Added ability to enter data into google sheets by having the entries as USER_ENTERED instead of RAW

Week 7:

Problems faced –
- Unicode characters in scarped data
- Manually retry if scrapping fails
- Finding the fulfillable PO orders by checking the warehouse inventory

Steps taken –
- Removed Unicode characters from the scrapped data
- A fully automatic scrapper retries(3 times total) if any pincodes fail repeatedly. 15 mins break between each retry
- Wrote a google app script to find out the number of fulfillable PO orders according to stock in the warehouse and the total revenue from those orders.