AN INDUSTRIAL TRAINING REPORT

At

Everything Beautiful Retail

Date: 9th June 2025 – 31st July 2025

SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENT FOR THE AWARD OF THE DEGREE OF

BACHELOR OF ENGINEERING
(COMPUTER SCIENCE AND ENGINEERING)

Submitted By:
Tushya Gupta (UE233106)
5th Semester

Department of Computer Science Engineering
University Institute of Engineering and Technology
Panjab University, Chandigarh
July, 2025

Table Of Contents

Acknowledgement

I would like to express my sincere gratitude to Everything Beautiful Retail for providing me with the opportunity to carry out my internship and gain practical experience in the field of e-commerce analytics. Working with the team gave me valuable exposure to real-world challenges in data collection, automation, and competitive analysis on Amazon, which greatly enhanced my technical and problem-solving skills.

I am deeply thankful to my project guide and the technical mentors at the company for their constant encouragement and expert guidance throughout the internship period. Their feedback and insights were instrumental in shaping the scrapers, dashboards, and analysis that form the core of this report.

My appreciation also extends to the entire operations and support staff for their cooperation and timely assistance in providing access to data, resources, and infrastructure whenever required. Their willingness to help made it possible to complete each stage of the project efficiently.

I am equally grateful to the colleagues and team members with whom I worked closely. Their collaborative spirit, discussions, and suggestions contributed greatly to improving the quality and scope of the work.

Finally, I sincerely acknowledge the support of everyone at Everything Beautiful Retail, directly or indirectly, whose contributions enabled the successful completion of this internship and enriched my professional learning experience.

Introduction

1.1 Company Profile

Everything Beautiful Retail is a dynamic e-commerce company specializing in a diverse array of consumer products, leveraging leading online platforms to reach a broad customer base, with Amazon serving as its primary marketplace. The company's core mission revolves around delivering high-quality products, ensuring customer satisfaction, and providing efficient and reliable delivery services. By maintaining a focus on quality and service, Everything Beautiful Retail has established a trusted reputation among its customers, positioning itself as a preferred choice in the highly competitive online retail space.

In addition to its primary retail operations, the company manages a specialized brand called BoxJoy, which offers curated product bundles and exclusive offerings aimed at enhancing convenience and creating unique customer experiences. BoxJoy is designed to cater to modern consumers seeking thoughtfully packaged products that combine practicality with a touch of personalization.

The company's growth strategy emphasizes the integration of strong retail expertise with modern digital marketplace tactics. By analysing market trends, monitoring consumer behaviour, and optimizing product listings, Everything Beautiful Retail has steadily expanded its online presence and cultivated a loyal customer base across India. The organization places significant emphasis on data-driven decision-making and technology adoption, employing analytics and emerging digital tools to guide inventory management, pricing strategies, marketing initiatives, and operational efficiency. This approach enables the company to respond proactively to shifts in consumer demand, strengthen brand identity, and enhance product visibility, ensuring it remains competitive in the fast-paced e-commerce ecosystem.

Through a combination of innovation, strategic planning, and customer-centric practices, Everything Beautiful Retail has successfully positioned itself as a forward-thinking e-commerce company that balances quality, convenience, and growth, reflecting a strong vision for sustainable success and long-term industry impact.

1.2 Project Description

This project involved the end-to-end design and implementation of automated systems to monitor and analyse product performance on Amazon, with the overarching goal of providing Everything Beautiful Retail with actionable, data-driven insights. The project aimed to give the company a clear understanding of how its products were performing across different regions, identify trends in customer behaviour, and benchmark performance against key competitors.

Over a seven-week period, a comprehensive suite of Python-based scrapers and supporting data pipelines was developed. These tools were designed to extract critical information such as product rankings, BuyBox ownership, delivery timelines, inventory levels, and order statuses. To ensure the system remained robust against Amazon's anti-bot measures, advanced techniques such as proxy rotation, browser impersonation, and built-in backup and retry mechanisms were implemented. This approach minimized downtime and maximized data accuracy, even under changing platform constraints.

Once collected, the data was processed and structured to enable efficient analysis. Google Sheets APIs were utilized for data storage and intermediate manipulation, while interactive dashboards were built using Looker Studio to visualize performance trends effectively. These dashboards allowed stakeholders to quickly interpret geographic variations in sales, monitor competitor activity, and track operational metrics over time. By integrating automated monitoring with intuitive visualization, the project provided a scalable solution that transformed raw e-commerce data into strategic insights, supporting decision-making for marketing, inventory management, and regional sales strategies.

Overall, this project not only strengthened the company's ability to respond to dynamic market conditions but also demonstrated the power of combining automation, data engineering, and visualization to drive measurable business impact in the digital retail space.

1.3 Scope of the Project

The primary objective of this project was to design and implement a comprehensive system for monitoring and analyzing product performance on Amazon, tailored to the specific needs of Everything Beautiful Retail. The project was structured into several interconnected components, each focusing on a key aspect of the data pipeline, analytics, and operational insight. Together, these components enabled the company to make informed, data-driven decisions regarding product strategy, inventory planning, and logistics optimization.

Data Collection

The foundation of the project was a robust data collection framework designed to capture critical metrics such as product rankings, BuyBox ownership, delivery timelines, and order status. The scrapers were modular, allowing for easy updates and maintenance as new products, categories, or regions were added. Special attention was given to fault tolerance, ensuring that intermittent errors, network failures, or website changes did not interrupt data collection. Data was gathered for all major Indian regions, enabling the company to identify geographic trends and regional variations in performance. By collecting granular information at the product and regional level, the system provided a detailed view of the competitive landscape, customer preferences, and fulfillment challenges.

Data Processing & Storage

Once collected, the raw data was processed and structured for efficient storage and retrieval. Outputs were integrated into Google Sheets and CSV pipelines, providing both secure storage and easy access for analysis and reporting. The processing pipeline included data cleaning, validation, and transformation steps to ensure consistency and accuracy, reducing the risk of erroneous insights. By standardizing data formats and implementing scalable storage solutions, the system was able to handle increasing volumes of data as the company expanded its product offerings and tracked additional metrics over time. This infrastructure also facilitated seamless integration with visualization and analytics tools, enabling the company to leverage insights quickly and effectively.

Visualization & Analysis

To make the collected data actionable, interactive dashboards were developed using Looker Studio. These dashboards presented key performance indicators such as delivery performance, product ranking distribution, competitor presence, and return patterns, broken down by state and pincode. Visualization was designed to be intuitive, allowing stakeholders to quickly identify trends, bottlenecks, and areas of opportunity without requiring technical expertise. Advanced filtering and drill-down capabilities were included to enable granular analysis, helping management and operational teams focus on specific products, regions, or performance metrics. These dashboards transformed raw data into clear, visual insights, supporting timely decision-making across marketing, sales, and operations teams.

Operational Insight

Beyond visualization, the system provided actionable operational insights to guide strategic decisions. Management gained real-time visibility into fulfillment bottlenecks, high-return products, and regions where competitors dominated. This allowed the company to optimize logistics, adjust pricing strategies, and plan inventory more effectively. By understanding performance patterns at a regional level, Everything Beautiful Retail was able to tailor its operational approach, such as prioritizing high-demand areas, improving delivery timelines where needed, and mitigating losses from high-return products. The insights generated also informed broader business strategy, including product launches, promotional campaigns, and competitor benchmarking.

Reliability & Scalability

Ensuring reliability and scalability was a critical aspect of the project. Amazon employs sophisticated anti-bot detection systems, and maintaining a continuous data flow required implementing strategies such as proxy rotation, CAPTCHA handling, and retry mechanisms. These measures ensured that scrapers remained operational even under changing platform constraints and minimized the risk of data loss or interruptions. The system was designed to scale with increasing data volumes, additional products, and expanding regions, allowing Everything Beautiful Retail to maintain a comprehensive and up-to-date view of its marketplace performance as the business grew.

Technologies Used

The internship required a diverse set of tools and technologies spanning web scraping, data engineering, visualization, and automation. Each component was chosen to address specific challenges in collecting and analysing large amounts of Amazon marketplace data.

I. Python and Core Libraries

Python was chosen as the primary programming language for this project because of its extensive ecosystem of mature libraries, readability, and rapid development capabilities. Its versatility allowed the team to prototype quickly, integrate multiple data sources, and deploy production-ready scrapers without excessive boilerplate code. Each stage of the project—from dynamic page rendering to network-level browser impersonation—relied on specialized Python libraries that worked together to form a robust, fault-tolerant scraping and data-processing pipeline.

1. Selenium – Browser Automation and Early Data Capture - At the outset, Selenium served as the key tool for interacting with Amazon's dynamic web pages. Because Amazon frequently uses JavaScript to render product details, simple HTTP requests were insufficient to capture complete page content. Selenium automated a real browser instance (in this case, headless Chrome) to simulate human interaction—loading pages, scrolling, and executing JavaScript. This approach ensured that session cookies and other authentication tokens were obtained legitimately, enabling early data collection and bypassing basic bot-detection mechanisms such as JavaScript challenge checks. Although Selenium added some overhead and was slower than pure HTTP requests, it provided a reliable starting point and a "human-like" footprint when the project was in its exploratory phase.

2. httpx – High-Performance Asynchronous Requests - As the scraper matured and the need for speed and scalability increased, the team migrated much of the data collection to httpx, a modern, Python-native HTTP client that supports asynchronous HTTP/2 requests. Unlike the traditional requests library, which operates synchronously and over HTTP/1.1, httpx allowed dozens of requests to be sent concurrently using Python's asyncio framework. This significantly reduced crawl times and server load while maintaining reliability. HTTP/2 multiplexing meant multiple data streams could share a single TCP connection, improving throughput and lowering latency—critical when tracking thousands of product URLs across different regions in near real time.

3. curl_cffi – Network-Layer Browser Impersonation - Even with Selenium and httpx, Amazon's sophisticated bot-protection systems occasionally flagged automated traffic. To address this, the project incorporated curl_cffi, a Python binding to libcurl that can mimic real browser network signatures at a very low level. By customizing TLS fingerprints, HTTP/2 settings, and header ordering, curl_cffi generated traffic that was virtually indistinguishable from that of Chrome or Firefox. This network-layer impersonation proved essential when Amazon began tightening detection heuristics; it enabled the scraper to continue operating without frequent CAPTCHAs or IP bans, ensuring uninterrupted data collection during high-volume crawls.

4. Parsel – Flexible HTML Parsing and Data Extraction (Appendix A) - After acquiring page responses, the next challenge was efficiently extracting structured data—such as product names, prices, rankings, seller details, and delivery estimates—from HTML that often varied by region and time. Parsel offered a unified solution with its powerful support for XPath, CSS selectors, and regular expressions. Its concise API allowed the team to write reusable extraction functions and quickly adapt to subtle markup changes. For example, a single Parsel selector could capture ranking information across multiple Amazon layouts, simplifying maintenance and reducing the likelihood of parsing errors.

II. Proxy and Network Management (Appendix B)

Because Amazon actively deploys sophisticated anti-bot defences—ranging from rate-limiting and behavioural analysis to TLS fingerprinting—robust proxy management became a core component of the scraping infrastructure. Without careful design, IP addresses would quickly be blacklisted, halting data collection and skewing analytics. The final system combined multiple layers of anonymity and traffic randomization to stay ahead of these controls.

1. Rotating Proxies for Load Distribution - A large pool of rotating residential and datacentre proxies served as the first line of defence. Each outgoing request was routed through a different IP address drawn from this pool, effectively distributing the crawl load across hundreds of endpoints. This rotation reduced the likelihood of triggering Amazon's rate-limit thresholds or geographic filters. Proxy selection incorporated randomization in timing, user-agent strings, and request patterns, simulating organic user traffic and making detection significantly harder.

2. TOR Network as a Backup Layer - When commercial proxies encountered localized blocking or temporary bans, the pipeline could seamlessly switch to the TOR network. TOR provided an extra layer of anonymity by routing traffic through a dynamic series of volunteer nodes, constantly changing the exit IP. Although TOR is slower than paid proxy services, it proved invaluable during high-scrutiny periods, acting as a "fallback mode" to keep the scrapers operational without interruption.

3. Low-Level Browser Impersonation - To complement IP rotation, the project employed curl_cffi to impersonate genuine browser behaviour at the network layer. Beyond simple header spoofing, curl_cffi allowed fine-grained control of TLS handshakes, cipher suites, and HTTP/2 settings, ensuring that each request's fingerprint closely matched that of real Chrome or Firefox traffic. This mitigated Amazon's more advanced detection techniques, which often rely on subtle differences in packet signatures rather than just IP addresses.

4. Coordinated Strategy - These three components—rotating proxies, TOR routing, and deep browser impersonation—worked in concert to create a multi-tiered defence. Requests could dynamically shift between proxy types based on response codes, latency, or detection signals. Combined with randomized request intervals and user-agent diversity, this architecture maintained a steady, reliable data flow even as Amazon adjusted its anti-scraping algorithms.

III. Data Handling and Automation

Capturing millions of rows of product rankings, BuyBox events, and delivery-time measurements demanded a storage architecture that was both reliable and collaborative. The system needed to ingest data at scale, safeguard it against network failures, and make it readily accessible for analysis and decision-making across teams.

1. Google Sheets API for Real-Time Collaboration $^{\text{(Appendix C)}}$ **-** At the heart of day-to-day reporting was the Google Sheets API, which enabled the scrapers to write data directly into shared spreadsheets without manual uploads. Using OAuth 2.0 authentication, the pipeline established secure, programmatic connections so that each scrape automatically refreshed dashboards and tables. This approach allowed managers and analysts to view near real-time updates, comment on anomalies, and download filtered views—all from a familiar spreadsheet interface. The ability to control permissions at a granular level ensured that sensitive data, such as pricing trends or competitive insights, was accessible only to authorized stakeholders.

2. High-Volume Exports with CSV Pipelines - However, Google Sheets enforces a 10-million-cell limit, which the project routinely approached when aggregating nationwide keyword rankings and hourly BuyBox snapshots. To handle this overflow, the system generated compressed CSV pipelines as a secondary storage layer. CSV files offered a lightweight, platform-agnostic format ideal for archival, offline analytics, and rapid ingestion into data-science environments such as pandas or Spark. Automated naming conventions and timestamped directories simplified version control, allowing historical trend analysis over months of data.

3. Resilient Backup and Retry Mechanisms **(Appendix D)** **-** Given the long-running nature of large scrapes—often spanning multiple days—resilience was critical. Custom Python scripts introduced checkpointing, recording the last successfully processed record so that, after a crash or temporary network outage, the scraper could resume from the exact failure point instead of restarting. Paired with automatic retry logic and exponential back-off, this design ensured that transient errors, proxy failures, or API timeouts did not jeopardize the completeness or integrity of the dataset.

4. Integrated Workflow for Scale and Reliability - Together, these components formed a multi-tiered data pipeline: Google Sheets for collaborative, real-time insights; CSV archives for high-volume and long-term storage; and robust backup routines for fault tolerance. This architecture not only preserved data accuracy and availability but also created a scalable foundation for future upgrades—such as migrating to cloud data warehouses like BigQuery or Redshift—without disrupting existing business processes.

IV. Visualization and Reporting

Transforming raw datasets into clear, actionable intelligence required a visualization layer that anyone—from marketing managers to operations leads—could navigate without technical training. To achieve this, the project relied heavily on Looker Studio, Google's modern rebranding of Data Studio, which offered the flexibility to build richly interactive dashboards directly on top of the continuously updated data feeds.

1. Looker Studio for Interactive Insights - Looker Studio served as the central hub for business intelligence, converting millions of rows of ranking data, BuyBox ownership logs, and delivery-timeline records into visually compelling reports. Using its native connectors, the system pulled from Google Sheets and CSV archives, then rendered geographical heatmaps and state- or pincode-level charts. These visual layers allowed stakeholders to instantly spot regions where product visibility lagged, competitors dominated, or delivery times fell below service benchmarks. Color-coded gradients—green for strong performance, amber for moderate, and red for urgent issues—provided an intuitive at-a-glance view of nationwide operations.

2. Custom Filters and Metrics - To empower decision-makers to drill into specific questions, the dashboards featured dynamic filters for sellers, delivery-time ranges, and individual keywords. Executives could, for example, select a single high-value keyword to compare the company's rank against competitors across different cities, or filter by a delivery window to evaluate the impact of logistics delays on search placement. Beyond standard charts, custom metrics—such as weighted click-through-rate estimates or BuyBox stability scores—were calculated directly within Looker Studio, giving teams immediate feedback without additional data-science tools.

3. Real-Time Collaboration and Strategic Value - Because Looker Studio operates entirely in the cloud, all visualizations updated in near real time as the underlying Google Sheets were refreshed by the scrapers. Multiple teams could view and comment simultaneously, ensuring that marketing, supply-chain, and executive stakeholders shared a single source of truth. This collaborative environment turned the dashboards into more than static reports; they became an interactive decision-support system, guiding advertising spend, inventory placement, and competitive strategy based on live marketplace conditions.

V. Amazon APIs and External Data Sources

To complement the front-end scraping infrastructure, the project leveraged official APIs and external market intelligence tools to provide a comprehensive view of product performance, customer behaviour, and market opportunities. Combining multiple data sources ensured accuracy, reduced gaps in information, and enabled more sophisticated analyses that were not possible using scraping alone.

1. Amazon Selling Partner API (SP-API) $^{\text{(Appendix E)}}$ **-** The Amazon Selling Partner API (SP-API) was a cornerstone for accessing official, back-end data. Unlike web scraping, which is limited to publicly available information, the SP-API provided structured and secure access to order histories, fulfilment statuses, return reports, and product-level revenue metrics. This allowed the team to validate and cross-reference scraper data against actual sales and shipping outcomes. For example, while a scraper could detect product rankings or BuyBox changes, SP-API data confirmed whether those changes translated into actual orders or revenue. Integration of SP-API outputs into the data pipelines also enabled automated reporting on fulfilment bottlenecks and returns, helping the operations team make data-driven decisions about logistics, inventory planning, and product improvement.

2. External Keyword Intelligence Tools - To further expand market insights, the project incorporated third-party keyword analysis platforms. These tools provided search volume estimates, click-through rates (CTR), and seasonal trend projections for high-value keywords, supplementing the historical and transactional data from SP-API. By combining internal ranking and order data with these external insights, the team could identify emerging high-demand keywords, optimize content and advertising campaigns, and discover untapped market opportunities. This multi-source approach enabled more accurate opportunity sizing and informed strategic decisions regarding product listings, promotional focus, and competitive positioning.

VI. Development and Packaging

Efficient software design and deployment were critical to ensure that the scraping and analytics system could be maintained, scaled, and used reliably across multiple machines. The project focused on creating a modular, portable architecture that allowed for both flexibility during development and simplicity during deployment.

1. Modular Python Structure - The codebase was organized into independent modules, each responsible for a specific aspect of the workflow: product ranking, BuyBox tracking, delivery timeline analysis, and order status monitoring. This modular structure provided multiple advantages. First, it simplified maintenance, allowing developers to update or debug a single module without affecting the rest of the system. Second, it facilitated scaling, as modules could be executed in parallel or on different servers to speed up data collection. Third, the separation of concerns made it easier to add new features, such as integrating additional marketplaces or advanced analytics, without overhauling the existing architecture.

2. Executable Packaging for Portability - To ensure that the system could be deployed across different machines and environments without requiring a full Python installation, the final scripts were packaged into standalone executables. This approach eliminated dependency issues and reduced setup time for new users. Special attention was given to path handling, so all input and output files—whether Google Sheets exports, CSV archives, or logs—were read and written relative to the executable's location. This design choice ensured that the system remained fully portable, could be run from different directories or machines, and allowed teams to easily schedule or automate tasks in diverse IT environments.

3. Benefits of the Approach - The combination of modular architecture and executable packaging created a robust, user-friendly system. Developers could continue to iterate on individual modules, analysts could rely on consistent and accurate data outputs, and operational staff could run scrapers or dashboards on multiple machines without requiring technical setup. Overall, this strategy maximized both flexibility during development and stability during deployment, laying a strong foundation for future expansion and integration of additional e-commerce data sources.

Workflow

These technologies worked together to create a robust data collection and visualization pipeline. Python libraries handled the heavy lifting of scraping and parsing, proxies and browser impersonation ensured access despite strict anti-bot defences, Google Sheets and CSV pipelines provided reliable storage, and Looker Studio transformed raw data into clear, actionable dashboards. Combined with Amazon's SP-API and external keyword tools, this technology stack delivered a complete solution for monitoring product rankings, analysing competition, and optimizing logistics on Amazon's marketplace.

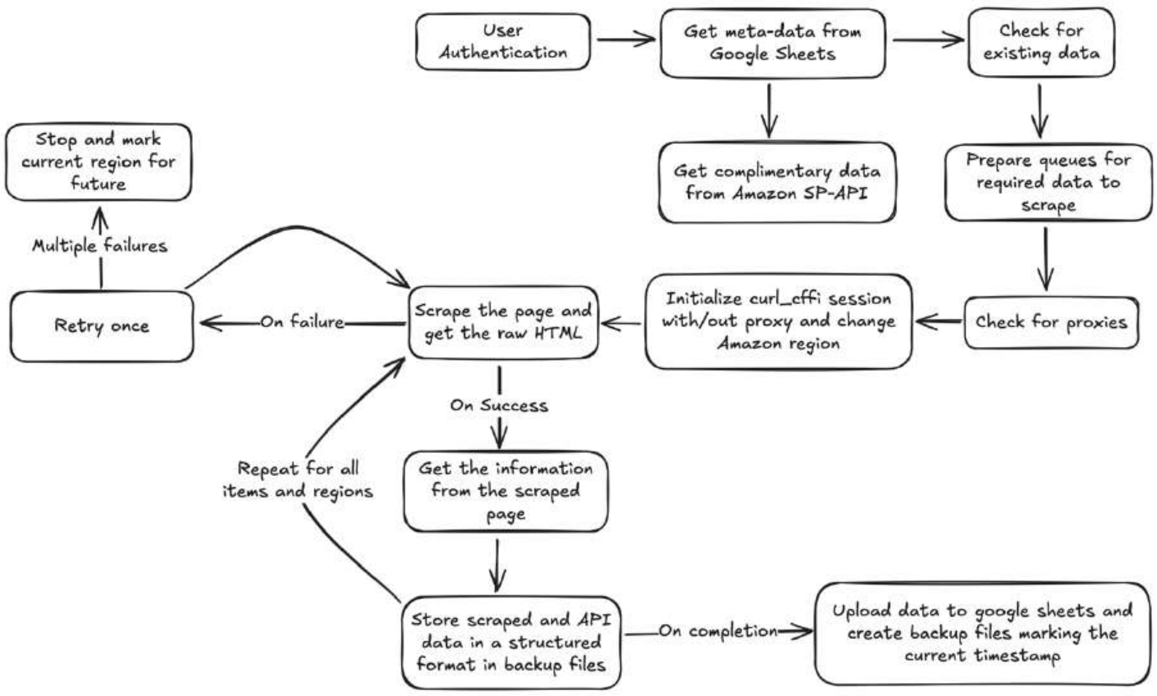

User Authentication
Get meta-data from Google Sheets
Check for existing data
Get complimentary data from Amazon SP-API
Prepare queues for required data to scrape
Stop and mark current region for future
Multiple failures
Retry once
On failure
Scrape the page and get the raw HTML
Initialize curl_cffi session with/out proxy and change Amazon region
Check for proxies
On Success
Repeat for all items and regions
Get the information from the scraped page
Store scraped and API data in a structured format in backup files
On completion
Upload data to google sheets and create backup files marking the current timestamp

Fig 1.1: Showing scraping workflow

Project Details

This internship project aimed to provide Everything Beautiful Retail and its brand *BoxJoy* with a comprehensive, data-driven understanding of how their products perform on Amazon across India. In a marketplace where visibility, delivery speed, and competitive pricing determine success, the company needed a reliable way to track key metrics—product rankings, BuyBox ownership, delivery timelines, keyword opportunities, and order statuses—in near real time. Over seven weeks, I built and refined a complete pipeline that automated data collection, storage, analysis, and visualization, turning millions of data points into actionable insights for management and operations teams.

Objectives

The primary objective of this project was to develop a comprehensive, automated system that provides Everything Beautiful Retail with actionable insights into product performance, customer engagement, and operational efficiency on Amazon. The project aimed to create a data-driven decision support system capable of monitoring key performance indicators across multiple dimensions, including product visibility, market competitiveness, fulfillment efficiency, and consumer behavior trends. By systematically collecting, processing, and visualizing large volumes of data, the system was intended to empower management and operational teams to make informed strategic decisions, optimize processes, and strengthen overall business performance.

Monitor Product Visibility $^{\text{(Appendix F)}}$: A fundamental objective of the project was to track and analyze product visibility across Amazon's platform. This involved monitoring ranking positions for selected keywords across hundreds of pincodes nationwide, creating a geographically detailed view of where products were performing well and where they were underperforming. Understanding geographic variations in visibility allowed the company to identify market gaps, regional demand patterns, and consumer preferences. These insights were critical for designing targeted marketing and promotional strategies, optimizing product listings, and allocating resources efficiently. By detecting trends in visibility over time, the system helped management anticipate shifts in customer behavior, respond proactively to competitive pressure, and ensure that high-priority products maintained strong presence across key regions.

Assess BuyBox Control $^{\text{(Appendix G)}}$: Another major objective focused on BuyBox ownership, which is directly linked to Amazon sales success. The system tracked which seller held the BuyBox and how frequently control changed, providing a detailed temporal and regional understanding of competitive dynamics. Monitoring BuyBox control enabled management to identify areas where competitors were gaining an advantage and allowed the company to implement strategies to regain or maintain dominance. This information informed critical operational and strategic decisions, including pricing adjustments, inventory allocation, and fulfillment prioritization. By maintaining a competitive edge in the BuyBox, Everything Beautiful Retail could maximize sales potential, strengthen market share, and enhance profitability across different product categories and regions.

Measure Delivery Performance: Delivery timelines were monitored extensively to evaluate fulfillment performance relative to competitors. By analyzing regional delivery performance, management could identify logistical bottlenecks, assess courier efficiency, and understand the impact of fulfillment speed on product rankings and customer satisfaction. This analysis supported operational improvements such as optimizing warehouse locations, prioritizing high-demand areas, and streamlining shipping processes. Additionally, correlating delivery performance with product visibility and BuyBox control helped the company understand how operational efficiency influences sales outcomes, enabling more informed investment in logistics infrastructure and service improvement initiatives.

Analyze Order Status and Returns: A key aspect of operational insight involved analyzing order status and return patterns. By identifying products with high return rates, frequent fulfillment issues, or delayed shipments, management could implement targeted corrective actions to reduce operational costs and enhance product quality. Tracking returns also provided insights into potential issues with packaging, shipping methods, or customer satisfaction. Understanding these trends allowed the company to anticipate and mitigate customer dissatisfaction, improve service reliability, and protect revenue. This objective was central to creating a proactive operational strategy that combined efficiency with quality control, reducing losses and strengthening customer trust.

Visualize Data for Decision-Making: Finally, the project aimed to transform complex datasets into intuitive, actionable visualizations that stakeholders could leverage without technical expertise. Interactive dashboards, geographic maps, and trend analyses allowed non-technical users to explore product performance metrics, delivery patterns, BuyBox fluctuations, and competitor activity. These visualizations made it easier to identify patterns, detect anomalies, and derive insights quickly, supporting faster and more informed decision-making. By presenting data in a clear and accessible format, the dashboards empowered management to align operational strategies, optimize marketing initiatives, and adapt to emerging market dynamics. In essence, visualization transformed raw data into a strategic asset, enabling the company to leverage insights for competitive advantage and long-term business growth.

Data Acquisition and Processing

The project began with the creation of robust Python-based scrapers designed to gather product and performance data from Amazon. The goal was to build a reliable, automated system capable of collecting both front-end marketplace data and back-end reports from Amazon's Selling Partner API (SP-API). The collected data would feed into processing pipelines and dashboards to generate actionable insights regarding product visibility, competitor performance, and operational efficiency.

Web Scraping Framework

The initial step involved developing a scraping framework capable of handling dynamic content on Amazon's website. Selenium was used to simulate real user interactions, navigate JavaScript-heavy pages, and initiate authenticated sessions. Selenium enabled the automation of tasks such as login, navigation to product pages, and capturing of dynamically loaded content. As the project evolved, the scraping pipeline was optimized for speed and stealth. The framework incorporated httpx for asynchronous HTTP/2 requests, significantly improving performance by allowing multiple requests to be handled concurrently. In addition, curl_cffi was utilized for browser impersonation at the network level, enabling scrapers to mimic real browser behavior and bypass detection mechanisms without the overhead of a full browser session. This hybrid approach balanced the need for accuracy, reliability, and efficiency in collecting large volumes of data.

Parsing Tools (Appendix A)

Once page content was captured, extracting meaningful information required a robust parsing strategy. Parsel was used to parse HTML, enabling the use of XPath, CSS selectors, and regular expressions for flexible and precise extraction of relevant data. This allowed the system to retrieve critical metrics such as product rankings, prices, customer ratings, and seller details. By designing modular parsers, the team ensured that the extraction process could be adapted quickly to changes in Amazon's front-end structure or for additional metrics. The parsing layer also included data validation and type conversion to ensure consistency across thousands of records collected daily.

Proxy and Anti-Detection Measures (Appendix B)
Amazon employs sophisticated anti-bot mechanisms to protect its platform from automated scraping. To mitigate this, multiple layers of proxy and anti-detection measures were implemented. Traffic was distributed across rotating proxies and TOR-based routing, ensuring that no single IP address was overloaded or flagged for suspicious activity. In addition, curl_cffi was configured to replicate real browser headers, TLS signatures, and other network characteristics to reduce the likelihood of detection. This careful configuration allowed the scrapers to maintain continuous operation while minimizing interruptions caused by IP bans, CAPTCHAs, or other defensive measures. These anti-detection strategies were critical for sustaining reliable, long-term data collection without manual intervention.

Backup and Retry Systems (Appendix D)
To improve robustness, the scrapers incorporated checkpointing and automatic retry mechanisms. Checkpointing allowed the system to resume from the last successfully completed step in case of a crash or interruption, minimizing data loss and avoiding the need to restart the scraping process from scratch. Automatic retries ensured that transient network errors, temporary page unavailability, or partial blocks did not disrupt data collection. This design enabled stable operation even under heavy blocking, server throttling, or network instability, allowing the team to collect comprehensive datasets over extended periods.

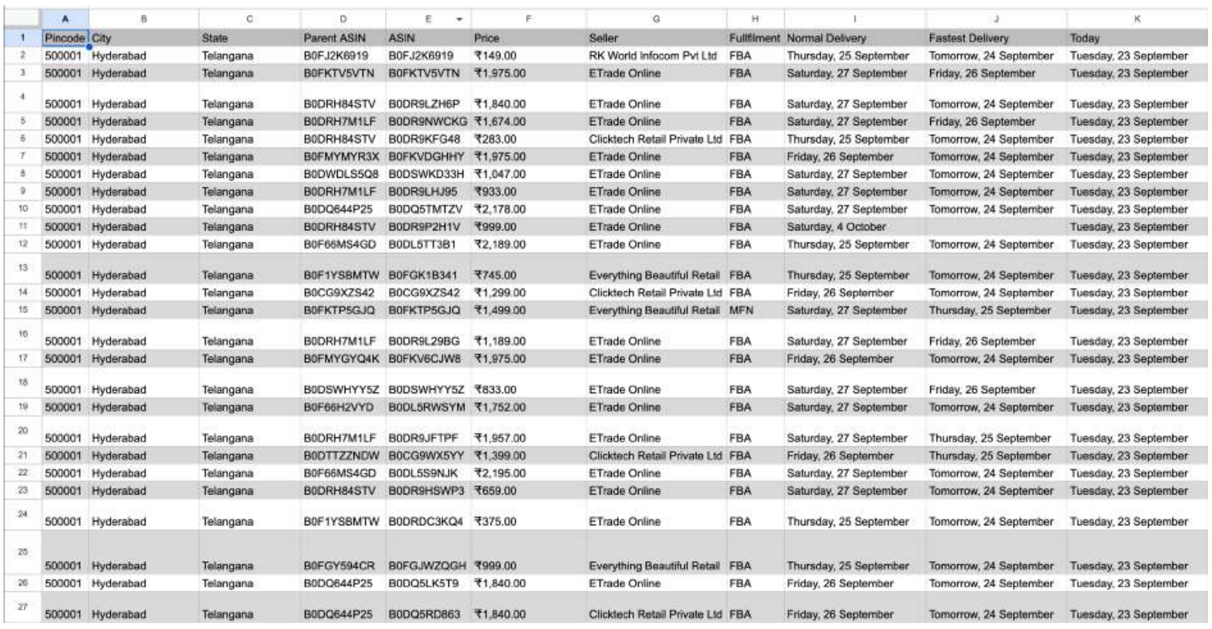
SP-API and External Data Integration (Appendix E)
For back-end data, the Amazon Selling Partner API (SP-API) provided structured reports including order statuses, return records, and keyword search volumes. These reports supplemented front-end scraped data, giving a complete view of product performance and operational metrics. To enrich insights, external keyword tools were integrated to estimate search volumes and click-through rates (CTR) for targeted keywords. This combination of front-end scraping, SP-API data, and external keyword metrics enabled a holistic view of both consumer demand and competitor activity.

End-to-End Pipeline Integration
All these components—web scraping, parsing, proxy management, retries, and SP-API integration—were tied together into a seamless data collection pipeline. Each layer was modular, allowing the team to quickly add new metrics or adjust strategies as Amazon's platform evolved. Data was then cleaned, validated, and fed into storage and visualization layers, forming the foundation for dashboards that enabled management to make informed, data-driven decisions.

Data Storage and Management

Efficient and secure storage of collected data was a key aspect of this project. The primary platform used was Google Sheets, which allowed real-time collaboration and simultaneous access for multiple team members. Integration with Google's OAuth 2.0 authentication ensured that automated updates were secure and properly authorized. To handle large volumes of data, scripts were designed to write results efficiently while maintaining compatibility across different machines, with files created relative to the executable's path to avoid manual reconfiguration.

When datasets approached Google Sheets' 10-million-cell limit, data was exported to CSV files for long-term archival and offline analysis. This provided a lightweight, platform-independent solution for historical storage and enabled further analysis with external tools. Incremental updates were handled efficiently to minimize redundant writes, ensuring that both Google Sheets and CSV datasets remained up-to-date. Overall, this approach combined real-time collaboration, secure automation, and scalable archival, providing a robust foundation for downstream analysis, visualization, and data-driven decision-making.



	A	B	C	D	E	F	G	H	I	J	K
1	Pincode	City	State	Parent ASIN	ASIN	Price	Seller	Fullfilment	Normal Delivery	Fastest Delivery	Today
2	500001	Hyderabad	Telangana	B0FJ2K6919	B0FJ2K6919	₹149.00	RK World Infocom Pvt Ltd	FBA	Thursday, 25 September	Tomorrow, 24 September	Tuesday, 23 September
3	500001	Hyderabad	Telangana	B0FKTV5VTN	B0FKTV5VTN	₹1,975.00	ETrade Online	FBA	Saturday, 27 September	Friday, 26 September	Tuesday, 23 September
4	500001	Hyderabad	Telangana	B0DRH84STV	B0DR9LZH6P	₹1,840.00	ETrade Online	FBA	Saturday, 27 September	Tomorrow, 24 September	Tuesday, 23 September
5	500001	Hyderabad	Telangana	B0DRH7M1LF	B0DR9NWCKG	₹1,674.00	ETrade Online	FBA	Saturday, 27 September	Friday, 26 September	Tuesday, 23 September
6	500001	Hyderabad	Telangana	B0DRH84STV	B0DR9KFG48	₹283.00	Clicktech Retail Private Ltd	FBA	Thursday, 25 September	Tomorrow, 24 September	Tuesday, 23 September
7	500001	Hyderabad	Telangana	B0FMYMYR3X	B0FKVDGHHY	₹1,975.00	ETrade Online	FBA	Friday, 26 September	Tomorrow, 24 September	Tuesday, 23 September
8	500001	Hyderabad	Telangana	B0DWDLS5Q8	B0DSWKD33H	₹1,047.00	ETrade Online	FBA	Saturday, 27 September	Tomorrow, 24 September	Tuesday, 23 September
9	500001	Hyderabad	Telangana	B0DRH7M1LF	B0DR9LHJ95	₹933.00	ETrade Online	FBA	Saturday, 27 September	Tomorrow, 24 September	Tuesday, 23 September
10	500001	Hyderabad	Telangana	B0DQ644P25	B0DQ5TMTZV	₹2,178.00	ETrade Online	FBA	Saturday, 27 September	Tomorrow, 24 September	Tuesday, 23 September
11	500001	Hyderabad	Telangana	B0DRH84STV	B0DR9P2H1V	₹999.00	ETrade Online	FBA	Saturday, 4 October		Tuesday, 23 September
12	500001	Hyderabad	Telangana	B0F66MS4GD	B0DL5TT3B1	₹2,189.00	ETrade Online	FBA	Thursday, 25 September	Tomorrow, 24 September	Tuesday, 23 September
13	500001	Hyderabad	Telangana	B0F1YSBMTW	B0FGK1B341	₹745.00	Everything Beautiful Retail	FBA	Thursday, 25 September	Tomorrow, 24 September	Tuesday, 23 September
14	500001	Hyderabad	Telangana	B0CG9XZS42	B0CG9XZS42	₹1,299.00	Clicktech Retail Private Ltd	FBA	Friday, 26 September	Tomorrow, 24 September	Tuesday, 23 September
15	500001	Hyderabad	Telangana	B0FKTP5GJQ	B0FKTP5GJQ	₹1,499.00	Everything Beautiful Retail	MFN	Saturday, 27 September	Thursday, 25 September	Tuesday, 23 September
16	500001	Hyderabad	Telangana	B0DRH7M1LF	B0DR9L29BG	₹1,189.00	ETrade Online	FBA	Saturday, 27 September	Friday, 26 September	Tuesday, 23 September
17	500001	Hyderabad	Telangana	B0FMYGYQ4K	B0FKV6CJW8	₹1,975.00	ETrade Online	FBA	Friday, 26 September	Tomorrow, 24 September	Tuesday, 23 September
18	500001	Hyderabad	Telangana	B0DSWHYY5Z	B0DSWHYY5Z	₹833.00	ETrade Online	FBA	Saturday, 27 September	Friday, 26 September	Tuesday, 23 September
19	500001	Hyderabad	Telangana	B0F66H2VYD	B0DL5RWSYM	₹1,752.00	ETrade Online	FBA	Saturday, 27 September	Tomorrow, 24 September	Tuesday, 23 September
20	500001	Hyderabad	Telangana	B0DRH7M1LF	B0DR9JFTPF	₹1,957.00	ETrade Online	FBA	Saturday, 27 September	Thursday, 25 September	Tuesday, 23 September
21	500001	Hyderabad	Telangana	B0DTTZZNDW	B0CG9WX5YY	₹1,399.00	Clicktech Retail Private Ltd	FBA	Friday, 26 September	Thursday, 25 September	Tuesday, 23 September
22	500001	Hyderabad	Telangana	B0F66MS4GD	B0DL5S9NJK	₹2,195.00	ETrade Online	FBA	Saturday, 27 September	Tomorrow, 24 September	Tuesday, 23 September
23	500001	Hyderabad	Telangana	B0DRH84STV	B0DR9HSWP3	₹659.00	ETrade Online	FBA	Saturday, 27 September	Tomorrow, 24 September	Tuesday, 23 September
24	500001	Hyderabad	Telangana	B0F1YSBMTW	B0DRDC3KQ4	₹375.00	ETrade Online	FBA	Thursday, 25 September	Tomorrow, 24 September	Tuesday, 23 September
25	500001	Hyderabad	Telangana	B0FGY594CR	B0FGJWZQGH	₹999.00	Everything Beautiful Retail	FBA	Thursday, 25 September	Tomorrow, 24 September	Tuesday, 23 September
26	500001	Hyderabad	Telangana	B0DQ644P25	B0DQ5LK5T9	₹1,840.00	ETrade Online	FBA	Friday, 26 September	Tomorrow, 24 September	Tuesday, 23 September
27	500001	Hyderabad	Telangana	B0DQ644P25	B0DQ5RD863	₹1,840.00	Clicktech Retail Private Ltd	FBA	Friday, 26 September	Tomorrow, 24 September	Tuesday, 23 September

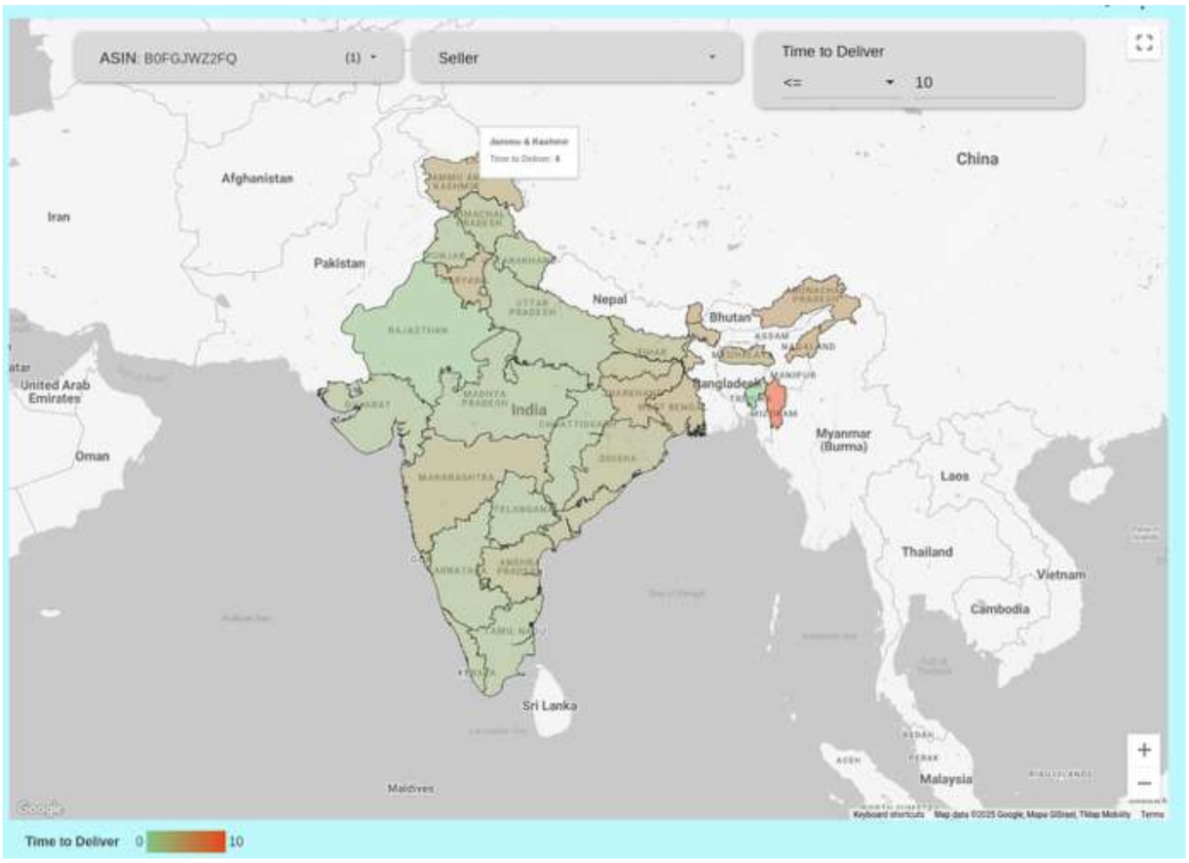
Fig 2.1: Google sheets showing scraped data

Visualization and Dashboarding

1. Delivery Timelines

One of the critical components of the dashboards was monitoring delivery performance across different regions. Using Looker Studio, delivery data was visualized as geographic heatmaps, where regions with slower delivery times were highlighted in red. This allowed management to quickly identify logistical bottlenecks and areas that required operational improvement. By analyzing trends over time, the company could detect recurring delays related to specific warehouses, courier partners, or regional challenges such as infrastructure limitations or peak-season surges.

Filters were implemented to allow stakeholders to view delivery metrics for individual products, sellers, or regions, providing granular insights. This enabled targeted interventions, such as optimizing shipping routes, adjusting inventory placement, or negotiating with logistics providers. By converting raw delivery data into interactive visualizations, the dashboards transformed complex operational information into actionable intelligence that directly supported decision-making and enhanced customer satisfaction.


ASIN: B0FGJWZ2FQ (1)

Seller

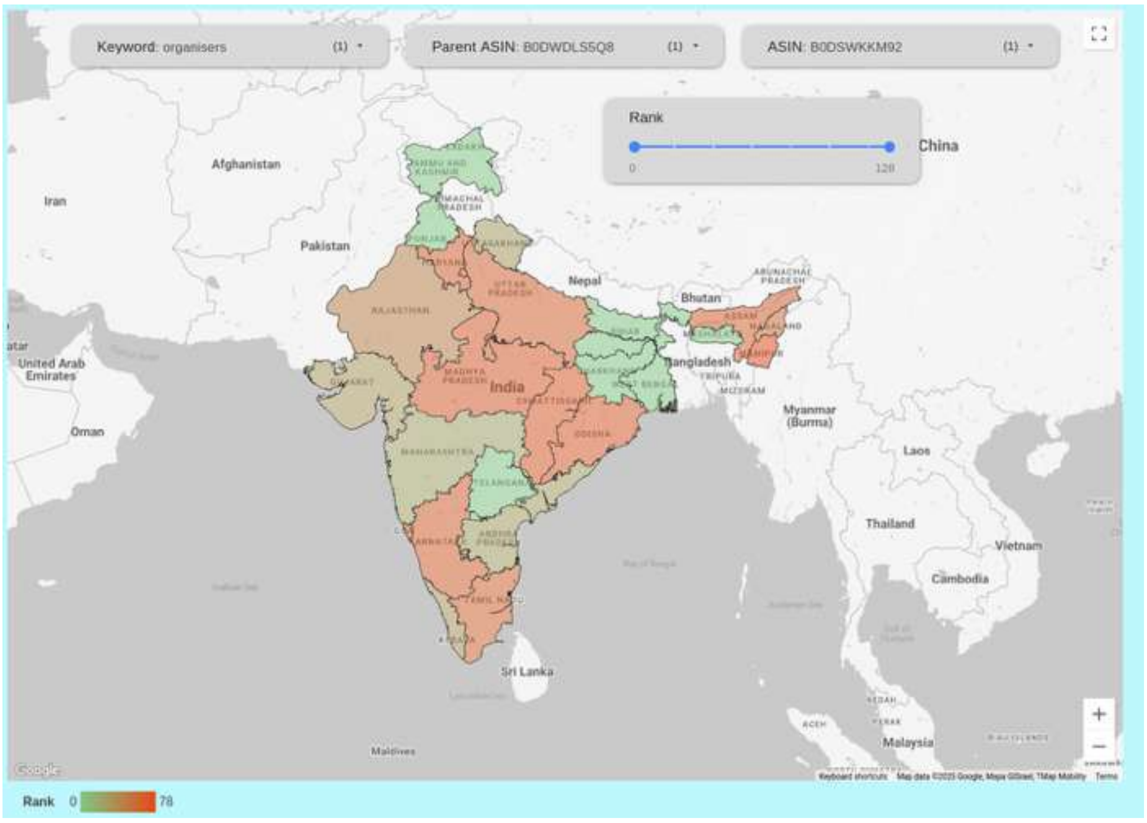
Time to Deliver <= 10

Fig 3.1: Map showing delivery timelines

2. Ranking Data

Tracking product performance in search rankings was another central focus of the dashboards. Ranking maps were created to provide a clear view of how products performed across different regions, color-coded to show strong visibility versus areas where performance lagged. These maps allowed the company to monitor trends in BuyBox ownership and identify geographic strongholds of competitors, offering insights into regional market dynamics.

Interactive features enabled filtering by product, seller, or keyword, allowing stakeholders to analyze specific aspects of performance in detail. By visualizing ranking fluctuations over time, management could understand the impact of promotions, price changes, or inventory availability on product visibility. These insights guided strategic marketing initiatives, pricing decisions, and product placement strategies, ensuring that the company could proactively respond to shifts in customer behavior and competitive activity.



Keyword: organisers (1)

Parent ASIN: B0DWDLS5Q8 (1)

ASIN: B0DSWKKM92 (1)

Rank 0 – 128

Fig 3.2: Map showing ranking for products

3. Search Volume

The dashboards also incorporated search volume and keyword data, providing insights into market demand and marketing opportunities. Weighted click-through rates (CTR) and keyword opportunity layers were overlaid on regional maps to highlight areas with the highest potential for customer engagement. This allowed marketing teams to focus their efforts on high-value regions and optimize campaigns for maximum impact.

In addition, search volume data was combined with delivery and ranking metrics to give a holistic view of product performance. By integrating multiple datasets, the dashboards revealed correlations between product visibility, customer interest, and fulfilment efficiency. This enabled informed decision-making around inventory allocation, promotional targeting, and regional expansion strategies. Automated refreshes ensured that the dashboards remained current, giving stakeholders near real-time intelligence to guide operational and strategic initiatives.

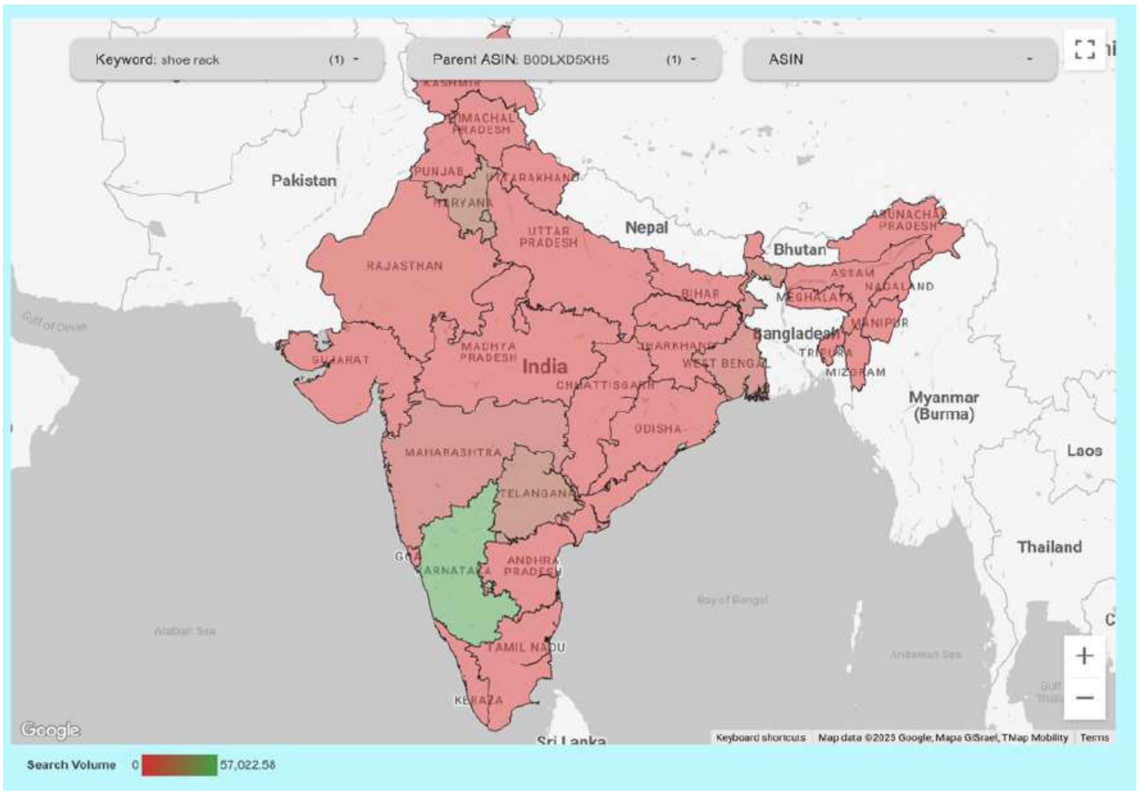


Fig 3.3: Map showing search volume

Data Analysis and Interpretation

The analysis of collected data provided Everything Beautiful Retail with actionable insights into product performance, customer behavior, and operational efficiency across India. By examining metrics such as product rankings, BuyBox ownership, delivery performance, search volumes, and order returns, the company was able to identify both strengths and areas for improvement. Each of these components revealed critical patterns that informed decision-making and helped shape marketing, inventory, and logistics strategies.

1. Product Rankings Across Regions
Analysis of product rankings revealed pronounced geographic variations. Products consistently performed best in metro cities such as Mumbai, Delhi, and Bangalore, where higher demand, better infrastructure, and concentrated consumer bases supported visibility and sales. In contrast, Tier-2 and Tier-3 cities showed a drop in rankings, suggesting that factors such as regional stock availability, lower search volumes, and heightened local competition influenced placement in search results. These insights highlighted the need for targeted regional strategies, including optimized inventory distribution and localized marketing efforts, to improve visibility in underperforming areas.

2. BuyBox Ownership and Seller Analysis
Monitoring BuyBox ownership revealed the competitive dynamics of the marketplace. Frequent changes in which seller held the BuyBox demonstrated how pricing strategies, stock availability, and fulfilment methods such as Fulfilled by Amazon (FBA) influenced the primary purchase option. By mapping these patterns over time, management could identify periods of vulnerability and design strategies to maintain or regain control. This insight informed tactical decisions regarding pricing adjustments, inventory replenishment, and promotional campaigns, ensuring that the company maximized its sales potential in each region.

3. Delivery Timeline and Service Coverage
Delivery performance was found to be a critical factor in maintaining high rankings and customer satisfaction. Analysis of delivery timelines showed that faster deliveries consistently correlated with more stable product rankings. Urban centres benefited from more reliable courier networks and warehouse proximity, while rural areas and smaller towns exhibited significant delays. These findings underscored the importance of logistics optimization, such as strategic warehouse placement, courier selection, and prioritization of high-demand regions, to ensure consistent service coverage and minimize disruptions to product visibility.

4. Search Volume and CTR Estimation
Examining search volume and estimated click-through rates (CTR) enabled the company to identify growth opportunities and maintain competitive advantage. High-volume keywords where the company had low visibility indicated untapped potential and areas where targeted marketing, keyword optimization, and advertising campaigns could drive incremental sales. Conversely, strong rankings in niche keywords represented areas where the company was already performing well and should continue to maintain dominance. Overlaying search data with geographic performance metrics allowed for the prioritization of high-value regions, ensuring marketing and inventory resources were allocated efficiently.

5. Order and Return Analysis
Analysis of backend order reports and returns provided insights into operational efficiency and product quality. High return rates highlighted specific products that required attention, whether through improved packaging, enhanced product descriptions, or adjustments in manufacturing. Mapping returns geographically revealed patterns that correlated with delivery methods, packaging robustness, and regional handling conditions. Additionally, analysing order statuses exposed fulfilment bottlenecks, enabling the company to streamline operations and reduce delays, ultimately improving both customer satisfaction and cost efficiency.

Key Findings

The combined scraping, backend integration, and dashboarding produced a rich set of findings that guided Everything Beautiful Retail's strategic planning and operational improvements. The key insights are outlined below, each offering a deeper understanding of market dynamics and internal performance.

Regional Delivery Disparities
Analysis of delivery timelines exposed significant regional performance gaps. Urban centres—particularly Delhi, Mumbai, and Bangalore—consistently benefited from fast, predictable fulfilment. Proximity to established Amazon warehouses and the availability of multiple courier partners meant that orders were often delivered within one or two days, reinforcing higher customer satisfaction and better product rankings. In contrast, customers in northern and northeastern states such as Assam, Manipur, and parts of Jammu & Kashmir experienced longer delivery windows and occasional delays. These delays stemmed from a combination of warehouse distance, limited last-mile logistics options, and challenging terrain. The data highlighted the business case for exploring new fulfilment partnerships or strategically placing inventory in additional regional hubs to ensure equitable service across India.

Competitor Dominance in Key Markets
Keyword and ranking data revealed that certain high-volume keywords and geographic regions were consistently dominated by entrenched competitors. Products from established national brands occupied top ranking positions, limiting Everything Beautiful Retail's visibility despite competitive pricing. This insight underscored the need for a dual strategy: aggressive advertising to challenge these competitors on high-traffic keywords and, simultaneously, focused campaigns on mid-tier keywords where incremental visibility gains could yield better returns on ad spend.

Emergence of New Sellers
The BuyBox monitoring tools detected the arrival of new market entrants offering similar product lines. Many of these sellers leveraged Fulfilled by Amazon (FBA) services, enabling them to provide rapid delivery and to benefit from Amazon's credibility indicators such as the Prime badge. Their sudden presence in competitive listings represented both a threat and an opportunity. Early detection allowed the company to track pricing tactics, inventory levels, and customer reviews of these newcomers, informing timely countermeasures such as limited-time discounts or bundling offers to defend market share.

Volatile BuyBox Control $^{\text{(Appendix G)}}$
Ownership of the BuyBox—the critical "Add to Cart" purchase option—proved highly volatile. In some categories, control shifted multiple times within a single day, driven largely by competitor price adjustments and temporary stockouts. This volatility had a direct impact on revenue, as losing the BuyBox significantly reduces conversion rates. Continuous monitoring allowed the team to correlate price changes with lost BuyBox periods and to implement automated alerts when thresholds were crossed. Management used these insights to fine-tune pricing strategies and maintain inventory buffers, ensuring more stable BuyBox retention.

Order and Return Insights
Backend order and return reports illuminated specific products with above-average return rates, pointing to potential quality control or packaging issues. In several cases, customer feedback cited discrepancies between product descriptions and actual items received, as well as damage during shipping. These findings prompted a review of product listings for clearer descriptions and the introduction of reinforced packaging for fragile items. Addressing these issues not only protected revenue but also helped maintain seller ratings, which are crucial for search visibility on Amazon.

Keyword Opportunity Sizing
The integration of Amazon search volume data with third-party keyword tools enabled precise opportunity mapping. High-volume keywords where company visibility was low emerged as prime candidates for new marketing campaigns or product variants. Conversely, analysis revealed stable, lower-volume niches where the company already ranked strongly. These niches represented areas where modest, cost-effective marketing could sustain dominance without significant additional spend. The ability to segment and prioritize keywords in this way provided a roadmap for both aggressive expansion and efficient resource allocation.

Visualization Impact
Finally, the use of Looker Studio dashboards transformed complex datasets into intuitive visuals, including geographic heatmaps, ranking timelines, and competitor overlays. Managers who were not deeply technical could interact with these dashboards through filters for product category, region, and timeframe, allowing them to make rapid, evidence-based decisions. Automated data refreshes ensured that executives always had up-to-date intelligence when planning pricing adjustments, inventory transfers, or marketing pushes. The visualizations effectively turned millions of raw data points into actionable intelligence that guided day-to-day and quarterly strategy.

Strategic Recommendations

Drawing on the key insights uncovered during analysis, a set of targeted strategies was proposed to strengthen Everything Beautiful Retail’s operational efficiency, market presence, and customer satisfaction. These recommendations move beyond broad guidance and focus on specific, actionable steps that management can implement in the near term while also supporting long-term growth.

1. Expand or Reposition Inventory to Reduce Regional Delivery Delays - Data showed that northern and northeastern states consistently experienced slower delivery times due to warehouse distance and limited last-mile logistics. To address this, the company should explore strategic inventory placement in or near these underserved regions. Options include partnering with third-party fulfilment centres, negotiating with Amazon to utilize additional Fulfilled by Amazon (FBA) nodes, or establishing small regional stocking points for high-demand items. Even a pilot program in one or two key locations could shorten delivery windows by a full day, improving customer satisfaction and stabilizing search rankings—both of which directly influence conversion rates. Long-term, this geographic rebalancing will also create resilience against regional disruptions such as weather events or courier strikes.

2. Strengthen Keyword Targeting and Advertising for High-Value Search Terms - Keyword analysis revealed numerous high-volume search terms where company products had weak visibility. To capture this unmet potential, management should develop a multi-tiered keyword strategy. At the top tier, high-traffic keywords can be targeted with Amazon Sponsored Products campaigns, backed by competitive bidding to break into the first page of search results. Mid-tier keywords—those with moderate volume but lower competition—offer a more cost-efficient path to improved visibility and should be optimized through enhanced product titles, bullet points, and backend search terms. Regular A/B testing of advertising copy and keyword mixes can further refine performance. This focused approach ensures that advertising spend yields measurable, profitable growth instead of being spread too thinly across low-impact terms.

3. Implement Continuous BuyBox Monitoring and Dynamic Pricing - Because BuyBox control directly drives the majority of Amazon sales, the company needs a real-time monitoring system that alerts staff whenever control is lost or pricing thresholds are crossed. Building on the scrapers already in place, the next step is to integrate automated notifications—via email, Slack, or internal dashboards—so the team can respond within minutes. Pairing these alerts with a dynamic pricing engine allows for automatic, rule-based price adjustments that remain competitive without eroding margins. Such a system would prevent revenue losses caused by sudden competitor discounts or stockouts and reduce the manual effort currently required to track BuyBox status.

4. Address High-Return Products Through Quality and Packaging Improvements - Return analysis highlighted specific products with above-average return rates, often tied to quality issues, inconsistent packaging, or ambiguous descriptions. The company should launch a targeted product-improvement initiative, beginning with supplier audits to verify material consistency and manufacturing standards. For fragile items, upgraded packaging—double-walled cartons, better cushioning, or tamper-evident seals—can reduce transit damage. Product listings should be updated with clearer photography, dimensional details, and care instructions to set accurate customer expectations. Reducing returns not only protects revenue but also helps maintain strong seller ratings, which directly influence Amazon search rankings and overall brand reputation.

5. Continue Enhancing Dashboards for Real-Time, Company-Wide Decision Support - The Looker Studio dashboards proved invaluable for translating millions of data points into intuitive visuals. To build on this success, the company should invest in continuous enhancement of these dashboards. Recommended next steps include integrating additional data sources such as advertising spend, customer review sentiment, and competitor price histories. Introducing user-level access controls and mobile-friendly interfaces would make the dashboards more accessible to teams in logistics, marketing, and management. Regular stakeholder feedback sessions can guide new features, ensuring that decision-makers always have the most relevant, real-time insights at their fingertips.

Implementing these strategies will allow Everything Beautiful Retail to transform raw data into sustainable competitive advantage. Faster regional delivery, sharper keyword targeting, proactive BuyBox defense, reduced return rates, and ever-improving analytics will together strengthen market position and customer trust—laying the groundwork for continued growth in an increasingly dynamic e-commerce environment.

Impact and Value

This project clearly demonstrated how the integration of advanced web-scraping techniques, fault-tolerant data pipelines, and dynamic visualization tools can evolve into a powerful decision-support system for modern e-commerce operations. By unifying these components into a single workflow, Everything Beautiful Retail moved beyond periodic, manual reporting and gained a near real-time understanding of its business landscape.

Operational Insight – Continuous data feeds gave management instant visibility into delivery performance, order status, and regional service gaps. Instead of waiting for end-of-month summaries, teams could detect fulfilment bottlenecks or courier delays as they emerged, enabling rapid adjustments in warehouse dispatching and last-mile logistics.

Competitive Intelligence – The monitoring framework exposed competitor behavior with unprecedented clarity. From identifying new market entrants using Fulfilled by Amazon to tracking sudden pricing changes that affected BuyBox ownership, the company now has an early-warning system for shifts in the marketplace, allowing proactive responses rather than reactive fixes.

Actionable Data for Strategy – Most importantly, the project translated raw numbers into clear recommendations that guided real business decisions. Insights on high-return products prompted packaging and quality upgrades; keyword and ranking analyses reshaped marketing priorities; and delivery-time disparities informed plans for regional inventory expansion.

Conclusion

The internship provided a comprehensive learning experience in the fields of web scraping, automation, and data-driven analysis for e-commerce. The central objective was to design and implement systems capable of monitoring Amazon product rankings, BuyBox ownership, delivery timelines, and order statuses across different regions of India. Over the seven-week period, multiple scrapers and analytical tools were developed, tested, and refined, each addressing a specific challenge faced by the company.

The work resulted in several tangible outcomes. Automated scrapers were successfully built to track rankings and BuyBox sellers, while integration with Google Sheets and Looker Studio allowed the visualization of delivery times, regional product performance, and competitor activity. Order status and return analysis scripts provided operational visibility into customer orders and highlighted areas of improvement in product quality and logistics. The findings showed clear patterns—delivery disparities across regions, competitor dominance in certain markets, and rising competition from new sellers—all of which offered practical insights for strategic decision-making.

At the same time, the project highlighted key limitations. CAPTCHAs, proxy reliability issues, dynamic website structures, and storage constraints presented challenges that required creative workarounds. The dependence on third-party APIs and external tools further emphasized the need for caution when scaling such systems. Despite these challenges, the methodology proved effective, and the outcomes demonstrated the value of combining scraping, automation, and visualization for actionable insights.

Overall, the internship strengthened technical proficiency in Python, APIs, and data visualization while also improving problem-solving, documentation, and project management skills. More importantly, it provided the company with tools and insights that can guide improvements in logistics, keyword optimization, pricing, and product differentiation. The project reaffirmed the importance of data-driven strategies in e-commerce and showcased how even small-scale technical interventions can lead to meaningful business outcomes.

Future Scope

The work completed during this internship establishes a strong, flexible foundation for Everything Beautiful Retail's long-term data and analytics capabilities. The current suite of Python scrapers, automated pipelines, and interactive dashboards already delivers real-time insight into product rankings, BuyBox control, and operational performance. However, e-commerce platforms—and Amazon in particular—evolve quickly, introducing new features, changing ranking algorithms, and tightening anti-scraping defenses. To remain competitive and to unlock the next level of intelligence, several forward-looking initiatives are recommended.

1. Advanced Logistics Optimization - Delivery speed is one of the most powerful levers influencing Amazon search rankings and customer satisfaction. Future work can integrate real-time warehouse feeds, courier tracking data, and regional demand signals directly into the analytics platform. Machine-learning models could forecast short-term demand by state or pincode, recommending dynamic inventory placement so that high-velocity products are stocked closer to customers before spikes occur. Such predictive logistics would shorten delivery windows in Tier-2 and Tier-3 cities, reduce last-mile costs, and strengthen the link between fulfillment performance and ranking stability.

2. Enhanced Keyword and Market Intelligence - The current keyword and ranking pipeline can grow into a full-scale keyword intelligence and market-tracking platform. Automated alerts could flag emerging high-volume keywords, competitor advertising pushes, or sudden seasonal trends, allowing the marketing team to reallocate ad spend in near real time. Incorporating natural-language processing (NLP) on customer reviews and Q&A data would surface hidden opportunities—new product ideas, underserved niches, and long-tail keywords that traditional tools overlook. This richer market intelligence would guide not only advertising strategy but also product development and merchandising.

3. Continuous BuyBox Monitoring with Dynamic Pricing - Because BuyBox ownership directly dictates conversion rates, a dedicated 24/7 monitoring microservice is the logical next step. Building on the current scrapers, this service would push instant notifications (via email, Slack, or SMS) whenever control changes or when competitor pricing crosses critical thresholds. When paired with dynamic pricing algorithms, the system could automatically recommend—or, with management approval, implement—price adjustments within predefined limits. This combination of real-time detection and algorithmic response would significantly reduce revenue loss from sudden competitor actions or temporary stockouts.

4. Product Quality and Return Prediction - The existing return analysis can evolve into a predictive analytics module. By combining historical return rates, customer feedback sentiment, and product metadata, machine-learning models could flag items at risk of high return rates before issues become widespread. Early detection would trigger pre-emptive interventions such as supplier audits, packaging redesigns, or more accurate product descriptions. This proactive approach would protect seller ratings, reduce refund costs, and improve customer trust.

5. Broader Competitor and Marketplace Coverage - Currently focused on Amazon India, the scraping framework can be extended to other marketplaces like Flipkart, Meesho, Snapdeal, or even international Amazon domains (e.g., Amazon US, UK). Cross-platform comparisons would reveal pricing gaps, product trends, and untapped regions, helping management decide where to expand next. Such a multi-market perspective would also provide early warnings about competitors who launch internationally before entering India.

6. Scalable Data Infrastructure - As data volumes grow, the current combination of Google Sheets and Looker Studio will eventually face performance and storage limits. A natural progression is to migrate to a cloud data warehouse such as BigQuery, Snowflake, or AWS Redshift. These platforms support streaming ingestion, near-real-time querying, and scalable storage measured in terabytes. Pairing them with a robust visualization layer like Tableau or Power BI would enable richer analytics, faster dashboards, and collaborative exploration across departments, all while maintaining data security and governance.

7. Robust Automation and Maintenance - Amazon frequently updates its site structure and anti-bot measures. To keep pace, the scraping infrastructure should be fortified with automated testing and a continuous integration/continuous deployment (CI/CD) pipeline. Containerization using Docker and orchestration with Kubernetes would simplify deployment, ensure consistent environments, and allow the system to scale automatically during peak data-collection periods. Scheduled self-diagnostics and alerting would further reduce downtime, ensuring that decision-makers always have access to fresh, accurate data.

Bibliography

- Selenium - https://www.selenium.dev/
- Httpx - https://www.python-httpx.org/
- Curl_cffi - https://github.com/lexiforest/curl_cffi
- Parsel - https://github.com/scrapy/parsel
- TOR - https://www.torproject.org/
- Google API Client - https://developers.google.com/workspace/docs/api/quickstart/python
- Amazon SP-API - https://developer-docs.amazon.com/sp-api

Appendix A

```python
from parsel import Selector


def get_info(selector: Selector, other_seller: Selector | None = None) -> dict:
    form = selector.css("form#addToCart")

    if form is not None:
        price = (form.css("span.a-offscreen::text").get() or "").strip()
        seller = (form.css("a#sellerProfileTriggerId::text").get() or "").strip()
        fullfilment_text = (form.css("div#fulfillerInfoFeature_feature_div span.a-size-small.a-color-tertiary::text").get() or "").strip()
        fullfilment = "MFN" if "delivered by" in fullfilment_text.lower() else "FBA"
        normal_delivery = (
            form.css(
                "div#mir-layout-DELIVERY_BLOCK-slot-PRIMARY_DELIVERY_MESSAGE_LARGE span.a-text-bold::text"
            ).get()
            or ""
        ).strip()
        fastest_delivery = (
            form.css(
                "div#mir-layout-DELIVERY_BLOCK-slot-SECONDARY_DELIVERY_MESSAGE_LARGE span.a-text-bold::text"
            ).get()
            or ""
        ).strip()
    else:
        price = ""
        seller = "Out of stock"
        fullfilment = ""
        normal_delivery = ""
        fastest_delivery = ""

    limited_time_deal = selector.css("div#dealBadge_feature_div span") and "Yes" or "No"

    parent_asin_link = selector.css("div#twister_feature_div a::attr(href)").get()
    if parent_asin_link is not None:
        parent_asin = (parent_asin_link.split("twister_")[1]).split("?")[0]
    else:
        parent_asin = ""

    learn_more_tags = ""
    i = 0
    while True:
        tag = selector.css(f"a#aspect-button-0-{i}::attr(aria-label)").get()
        i += 1
        if tag is not None:
            learn_more_tags += tag + " - "
        else:
            break
    if learn_more_tags != "":
        learn_more_tags = (
            learn_more_tags[:-3]
            .replace("Positive aspect", "✓")
            .replace("Negative aspect", "X")
        )

    breadcrums = "".join(
        selector.css("div#desktop-breadcrumbs_feature_div ::text").getall()
    ).strip()
```

```python
rating_sub_division = selector.css(
    "ul#histogramTable div.a-section.a-spacing-none.a-text-right.aok-nowrap::text"
).getall()
if rating_sub_division is not None and len(rating_sub_division) > 4:
    star5 = rating_sub_division[0]
    star4 = rating_sub_division[1]
    star3 = rating_sub_division[2]
    star2 = rating_sub_division[3]
    star1 = rating_sub_division[4]
else:
    star5 = ""
    star4 = ""
    star3 = ""
    star2 = ""
    star1 = ""

best_seller1 = ""
best_seller2 = ""
best_seller3 = ""
ul = selector.css(
    "table#productDetails_detailBullets_sections1 ul.a-unordered-list.a-nostyle.a-vertical"
)
entries = ul.css("li")

best_seller = [entry.css("::text").getall() for entry in entries[1:]]
best_seller = ["".join(item).strip() for item in best_seller]

if len(best_seller) >= 1:
    best_seller1 = best_seller[0]
if len(best_seller) >= 2:
    best_seller2 = best_seller[1]
if len(best_seller) >= 3:
    best_seller3 = best_seller[2]

stars = (
    selector.css(
        "div#averageCustomerReviews span.a-size-base.a-color-base::text"
    ).get()
    or ""
)
total_ratings = (
    selector.css(
        "div#averageCustomerReviews span#acrCustomerReviewText::text"
    ).get()
    or ""
)

other_sellers = ""
num_other_sellers = 0
if other_seller is not None:
    # Select the offer list container
    offers = other_seller.css("div#aod-offer-list > div > div")
    for offer in offers:
        offer_seller = offer.css("a.a-size-small.a-link-normal::text").get() or ""
        offer_fullfilment_text = offer.css("div#aod-offer-shipsFrom span.a-size-small.a-color-tertiary::text").get() or ""
        offer_fullfilment = "MFN" if "delivered by" in offer_fullfilment_text.lower() else "FBA"
        offer_price = (
            "".join(
                offer.css(
                    "span.a-price.aok-align-center.centralizedApexPricePriceToPayMargin ::text"
                ).getall()
            )
            .strip()
        )

        offer_normal_delivery = (
            offer.css(
                "div#mir-layout-DELIVERY_BLOCK-slot-PRIMARY_DELIVERY_MESSAGE_LARGE span.a-text-bold::text"
            ).get()
            or ""
        )

        offer_fastest_delivery = (
            offer.css(
                "div#mir-layout-DELIVERY_BLOCK-slot-SECONDARY_DELIVERY_MESSAGE_LARGE span.a-text-bold::text"
            ).get()
            or ""
        )

        other_sellers += f"{offer_seller.strip()} - {offer_fullfilment.strip()} - {offer_price.strip()} - {offer_normal_deliver
        num_other_sellers += 1

    other_sellers = other_sellers.strip()

obj = {
    "price": price,
    "seller": seller,
    "fullfilment": fullfilment,
    "normal_delivery": normal_delivery,
    "fastest_delivery": fastest_delivery,
    "best_seller1": best_seller1,
    "best_seller2": best_seller2,
    "best_seller3": best_seller3,
    "stars": stars,
    "total_ratings": total_ratings,
    "limited_time_deal": limited_time_deal,
    "learn_more_tags": learn_more_tags,
    "star5": star5,
    "star4": star4,
    "star3": star3,
    "star2": star2,
    "star1": star1,
    "breadcrums": breadcrums,
    "parent_asin": parent_asin,
    "num_other_sellers": num_other_sellers,
    "other_sellers": other_sellers,
}
return obj
```

Appendix B

```
def multi_thread(
    pincode: str,
    proxy_type: str,
    result: queue.Queue,
    killed: queue.Queue,
    asins: dict,
    progress=None,
    task=None,
):
    if len(asins[pincode]) == 0:
        return
    print(f"Starting {pincode}")
    USER_AGENT = USER_AGENTS[random.randint(0, len(USER_AGENTS) - 1)]
    REQUEST_HEADERS = {"Accept-Language": "en", "User-Agent": USER_AGENT}

    if proxy_type == "None":
        client = Session(impersonate="firefox")
    elif proxy_type == "Single":
        proxy = proxies_single[pincode]
        client = Session(impersonate="firefox", proxy=proxy)
    elif proxy_type == "Multi":
        proxy = proxies_multi[pincode]
        client = Session(impersonate="firefox", proxy=proxy)
    else:
        client = Session(impersonate="firefox")

    res = get_session_cookies(
        client, str(pincode), USER_AGENT=USER_AGENT, REQUEST_HEADERS=REQUEST_HEADERS
    )
    if not res:
        killed.put(pincode)
        print(f"Killing {pincode}")
        return
    for asin in asins[str(pincode)]:
        time.sleep(0.5 + random.random() / 2)
        scrape(client, pincode, asin, result, REQUEST_HEADERS)
        if progress is not None:
            progress.advance(task, advance=1)
    print(f"Ending {pincode}")
```

Appendix C

```python
from googleapiclient.discovery import build
from googleapiclient.errors import HttpError

from google.get_creds import get_creds
from settings.config import sheet_id


def get_values(range_name, read_sheet_id=None, use_service_account=False):
    if read_sheet_id is None:
        read_sheet_id = sheet_id
    creds = get_creds(use_service_account)
    # pylint: disable=maybe-no-member
    try:
        service = build("sheets", "v4", credentials=creds)

        result = (
            service.spreadsheets()
            .values()
            .get(
                spreadsheetId=read_sheet_id,
                range=range_name,
                valueRenderOption="UNFORMATTED_VALUE",  # or FORMATTED_VALUE
                majorDimension="ROWS",  # default, can also be "COLUMNS"
            )
            .execute()
        )
        # rows = result.get("values", [])
        # print(f"{len(rows)} rows retrieved")
        return result
    except HttpError as error:
        print(f"An error occurred: {error}")
        return None
```

```python
from googleapiclient.discovery import build
from googleapiclient.errors import HttpError

from google.get_creds import get_creds
from settings.config import sheet_id


def get_sheet_id(write_sheet_id, service, sheet_name):
    metadata = service.spreadsheets().get(spreadsheetId=write_sheet_id).execute()
    for sheet in metadata["sheets"]:
        if sheet["properties"]["title"] == sheet_name:
            return sheet["properties"]["sheetId"]
    raise Exception(f"Sheet '{sheet_name}' not found.")


def update_values(
    range_name, values, write_sheet_id=None, shift_old=True, user_entered=False, use_service_account=False
):
    creds = get_creds(use_service_account)
    if write_sheet_id is None:
        write_sheet_id = sheet_id

    try:
        service = build("sheets", "v4", credentials=creds)
        if shift_old:
            insert_request = {
                "insertDimension": {
                    "range": {
                        "sheetId": get_sheet_id(
                            write_sheet_id, service, range_name.split("!")[0]
                        ),
                        "dimension": "COLUMNS",
                        "startIndex": 0,
                        "endIndex": len(values[0]) + 1,
                    },
                    "inheritFromBefore": False,
                }
            }
            body = {"requests": [insert_request]}
            service.spreadsheets().batchUpdate(
                spreadsheetId=write_sheet_id, body=body
            ).execute()
        else:
            clear_request = {
                "updateCells": {
                    "range": {
                        "sheetId": get_sheet_id(
                            write_sheet_id, service, range_name.split("!")[0]
                        ),
                        "startRowIndex": 0,
                        "startColumnIndex": 0,
                        "endColumnIndex": len(values[0]) + 1,
                    },
                    "fields": "userEnteredValue",
                }
            }
            body = {"requests": [clear_request]}
            service.spreadsheets().batchUpdate(
                spreadsheetId=write_sheet_id, body=body
            ).execute()

        data = [
            {"range": range_name, "values": values},
        ]

        if user_entered:
            body = {"valueInputOption": "USER_ENTERED", "data": data}
        else:
            body = {"valueInputOption": "RAW", "data": data}

        result = (
            service.spreadsheets()
            .values()
            .batchUpdate(spreadsheetId=write_sheet_id, body=body)
            .execute()
        )
        print(f"{(result.get('totalUpdatedCells'))} cells updated.")
        return result
    except HttpError as error:
        print(f"An error occurred: {error}")
        return error
```

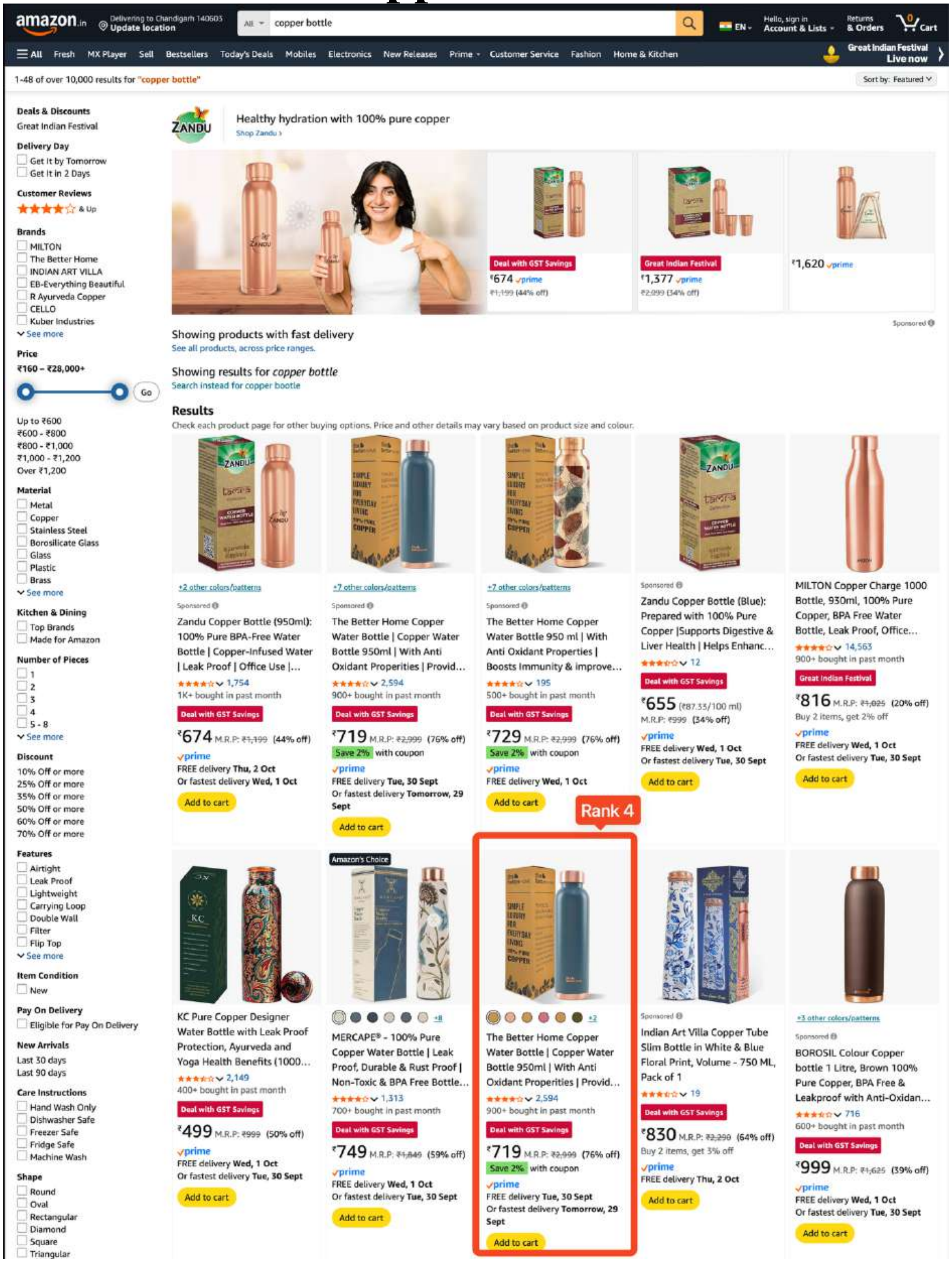
Appendix D

```python
loops = 0
while len(do_to_pincodes) != 0 and loops < 2:
    if loops != 0:
        print(f"Retrying {json.dumps(do_to_pincodes, indent=2)}")
    try:
        for i, pincode in enumerate(do_to_pincodes):
            with Progress(
                TextColumn("[bold blue]{task.description}"),
                BarColumn(),
                "[progress.percentage]{task.percentage:>3.0f}%",
                TimeElapsedColumn(),
                TimeRemainingColumn(),
            ) as progress:
                task = progress.add_task(
                    f"({(i + 1)}/{len(do_to_pincodes)}){pincode} ",
                    total=len(asins[pincode]),
                )
                multi_thread(
                    pincode, proxy_type, result, killed, asins, progress, task
                )
                progress.stop_task(task)
                with open(os.path.join(APP_DIR, "buybox.txt"), "w") as w:
                    w.write(json.dumps(list(result.queue), indent=2))
    except Exception as e:
        print(e)
    finally:
        with open(os.path.join(APP_DIR, "buybox.txt"), "w") as w:
            w.write(json.dumps(list(result.queue), indent=2))
    do_to_pincodes = list(killed.queue)
    while not killed.empty():
        killed.get()
    loops += 1
```

Appendix E

```python
def get_report_id(asins):
    url = "https://sellingpartnerapi-eu.amazon.com/reports/2021-06-30/reports"

    startDate, endDate = get_dates()

    payload = {
        "marketplaceIds": ["A21TJRUUN4KGV"],
        "reportType": "GET_BRAND_ANALYTICS_SEARCH_QUERY_PERFORMANCE_REPORT",
        "dataStartTime": startDate,
        "dataEndTime": endDate,
        "reportOptions": {
            "reportPeriod": "MONTH",
            "asin": asins,
        },
    }

    create_report_response = requests.post(url, headers=headers, json=payload)
    if create_report_response.status_code == 429:
        print("Rate limiting")
        time.sleep(60)
        create_report_response = requests.post(url, headers=headers, json=payload)
        create_report_response.raise_for_status()

    report_id = create_report_response.json()["reportId"]
    return report_id
```

Appendix F



amazon.in Delivering to Chandigarh 140603 Update location | All | copper bottle | EN | Hello, sign in Account & Lists | Returns & Orders | 0 Cart

All | Fresh | MX Player | Sell | Bestsellers | Today's Deals | Mobiles | Electronics | New Releases | Prime | Customer Service | Fashion | Home & Kitchen | Great Indian Festival Live now

1-48 of over 10,000 results for "copper bottle" | Sort by: Featured

Deals & Discounts
Great Indian Festival

Delivery Day
Get It by Tomorrow
Get It in 2 Days

Customer Reviews
★★★★☆ & Up

Brands
MILTON
The Better Home
INDIAN ART VILLA
EB-Everything Beautiful
R Ayurveda Copper
CELLO
Kuber Industries
See more

Price
₹160 – ₹28,000+
Go
Up to ₹600
₹600 - ₹800
₹800 - ₹1,000
₹1,000 - ₹1,200
Over ₹1,200

Material
Metal
Copper
Stainless Steel
Borosilicate Glass
Glass
Plastic
Brass
See more

Kitchen & Dining
Top Brands
Made for Amazon

Number of Pieces
1
2
3
4
5 - 8
See more

Discount
10% Off or more
25% Off or more
35% Off or more
50% Off or more
60% Off or more
70% Off or more

Features
Airtight
Leak Proof
Lightweight
Carrying Loop
Double Wall
Filter
Flip Top
See more

Item Condition
New

Pay On Delivery
Eligible for Pay On Delivery

New Arrivals
Last 30 days
Last 90 days

Care Instructions
Hand Wash Only
Dishwasher Safe
Freezer Safe
Fridge Safe
Machine Wash

Shape
Round
Oval
Rectangular
Diamond
Square
Triangular

ZANDU Healthy hydration with 100% pure copper
Shop Zandu ›

Deal with GST Savings ₹674 prime ~~₹1,199~~ (44% off)

Great Indian Festival ₹1,377 prime ~~₹2,099~~ (34% off)

₹1,620 prime

Sponsored

Showing products with fast delivery
See all products, across price ranges.

Showing results for *copper bottle*
Search instead for copper bootle

Results
Check each product page for other buying options. Price and other details may vary based on product size and colour.

+2 other colors/patterns
Sponsored
Zandu Copper Bottle (950ml): 100% Pure BPA-Free Water Bottle | Copper-Infused Water | Leak Proof | Office Use |...
★★★★☆ 1,754
1K+ bought in past month
Deal with GST Savings
₹674 M.R.P: ~~₹1,199~~ (44% off)
prime
FREE delivery **Thu, 2 Oct**
Or fastest delivery **Wed, 1 Oct**
Add to cart

+7 other colors/patterns
Sponsored
The Better Home Copper Water Bottle | Copper Water Bottle 950ml | With Anti Oxidant Properities | Provid...
★★★★☆ 2,594
900+ bought in past month
Deal with GST Savings
₹719 M.R.P: ~~₹2,999~~ (76% off)
Save 2% with coupon
prime
FREE delivery **Tue, 30 Sept**
Or fastest delivery **Tomorrow, 29 Sept**
Add to cart

+7 other colors/patterns
Sponsored
The Better Home Copper Water Bottle 950 ml | With Anti Oxidant Properties | Boosts Immunity & improve...
★★★★☆ 195
500+ bought in past month
Deal with GST Savings
₹729 M.R.P: ~~₹2,999~~ (76% off)
Save 2% with coupon
prime
FREE delivery **Wed, 1 Oct**
Add to cart

Sponsored
Zandu Copper Bottle (Blue): Prepared with 100% Pure Copper |Supports Digestive & Liver Health | Helps Enhanc...
★★★☆☆ 12
Deal with GST Savings
₹655 (₹87.33/100 ml)
M.R.P: ~~₹999~~ (34% off)
prime
FREE delivery **Wed, 1 Oct**
Or fastest delivery **Tue, 30 Sept**
Add to cart

MILTON Copper Charge 1000 Bottle, 930ml, 100% Pure Copper, BPA Free Water Bottle, Leak Proof, Office...
★★★★☆ 14,563
900+ bought in past month
Great Indian Festival
₹816 M.R.P: ~~₹1,025~~ (20% off)
Buy 2 items, get 2% off
prime
FREE delivery **Wed, 1 Oct**
Or fastest delivery **Tue, 30 Sept**
Add to cart

Rank 4

KC Pure Copper Designer Water Bottle with Leak Proof Protection, Ayurveda and Yoga Health Benefits (1000...
★★★☆☆ 2,149
400+ bought in past month
Deal with GST Savings
₹499 M.R.P: ~~₹999~~ (50% off)
prime
FREE delivery **Wed, 1 Oct**
Or fastest delivery **Tue, 30 Sept**
Add to cart

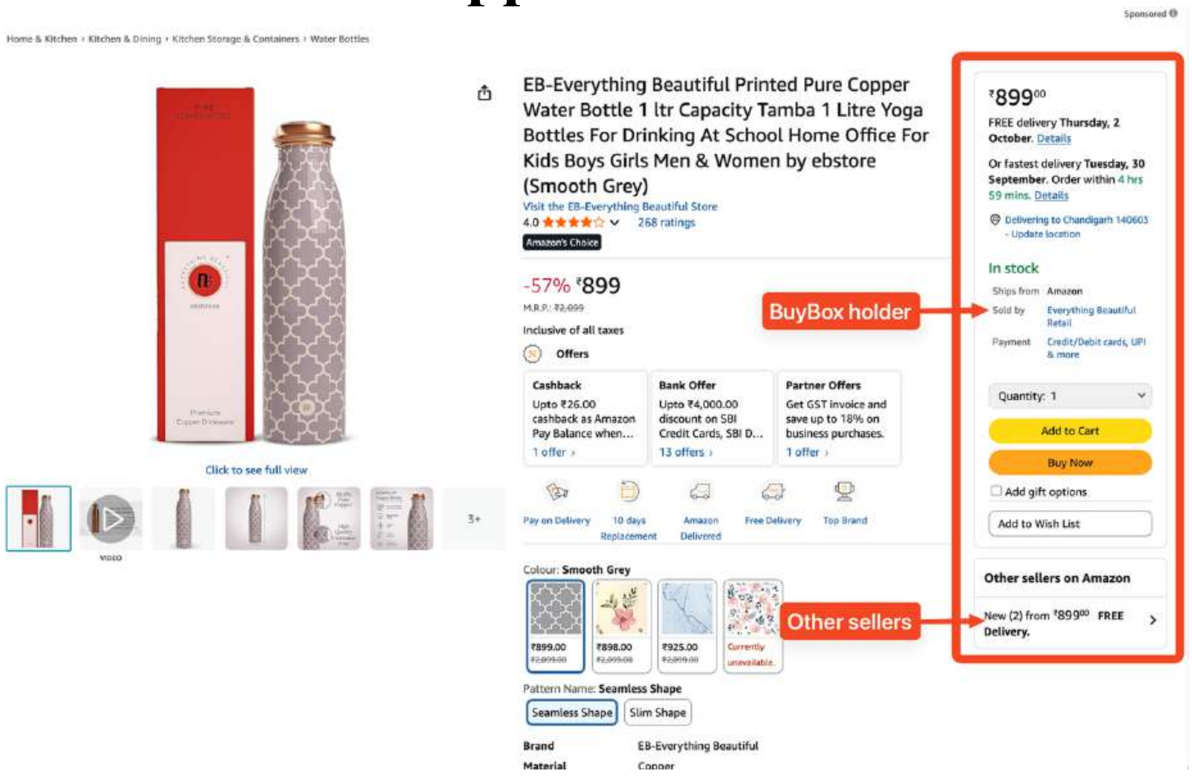
Amazon's Choice
+8
MERCAPE® - 100% Pure Copper Water Bottle | Leak Proof, Durable & Rust Proof | Non-Toxic & BPA Free Bottle...
★★★★☆ 1,313
700+ bought in past month
Deal with GST Savings
₹749 M.R.P: ~~₹1,849~~ (59% off)
prime
FREE delivery **Wed, 1 Oct**
Or fastest delivery **Tue, 30 Sept**
Add to cart

+2
The Better Home Copper Water Bottle | Copper Water Bottle 950ml | With Anti Oxidant Properities | Provid...
★★★★☆ 2,594
900+ bought in past month
Deal with GST Savings
₹719 M.R.P: ~~₹2,999~~ (76% off)
Save 2% with coupon
prime
FREE delivery **Tue, 30 Sept**
Or fastest delivery **Tomorrow, 29 Sept**
Add to cart

Sponsored
Indian Art Villa Copper Tube Slim Bottle in White & Blue Floral Print, Volume - 750 ML, Pack of 1
★★★☆☆ 19
Deal with GST Savings
₹830 M.R.P: ~~₹2,290~~ (64% off)
Buy 2 items, get 3% off
prime
FREE delivery **Thu, 2 Oct**
Add to cart

+3 other colors/patterns
Sponsored
BOROSIL Colour Copper bottle 1 Litre, Brown 100% Pure Copper, BPA Free & Leakproof with Anti-Oxidan...
★★★☆☆ 716
600+ bought in past month
Deal with GST Savings
₹999 M.R.P: ~~₹1,625~~ (39% off)
prime
FREE delivery **Wed, 1 Oct**
Or fastest delivery **Tue, 30 Sept**
Add to cart

Appendix G



Sponsored

Home & Kitchen › Kitchen & Dining › Kitchen Storage & Containers › Water Bottles

EB-Everything Beautiful Printed Pure Copper Water Bottle 1 ltr Capacity Tamba 1 Litre Yoga Bottles For Drinking At School Home Office For Kids Boys Girls Men & Women by ebstore (Smooth Grey)

Visit the EB-Everything Beautiful Store

4.0 ★★★★☆ 268 ratings

Amazon's Choice

-57% ₹899

M.R.P.: ~~₹2,099~~

Inclusive of all taxes

Offers

Cashback	Bank Offer	Partner Offers
Upto ₹26.00 cashback as Amazon Pay Balance when…	Upto ₹4,000.00 discount on SBI Credit Cards, SBI D…	Get GST invoice and save up to 18% on business purchases.
1 offer ›	13 offers ›	1 offer ›

Pay on Delivery · 10 days Replacement · Amazon Delivered · Free Delivery · Top Brand

Colour: Smooth Grey

₹899.00 ~~₹2,099.00~~	₹898.00 ~~₹2,099.00~~	₹925.00 ~~₹2,099.00~~	Currently unavailable.

Pattern Name: Seamless Shape

Seamless Shape | Slim Shape

Brand EB-Everything Beautiful

Material Copper

₹899.00

FREE delivery Thursday, 2 October. Details

Or fastest delivery Tuesday, 30 September. Order within 4 hrs 59 mins. Details

Delivering to Chandigarh 140603 - Update location

In stock

Ships from Amazon

Sold by Everything Beautiful Retail

Payment Credit/Debit cards, UPI & more

Quantity: 1

Add to Cart

Buy Now

Add gift options

Add to Wish List

Other sellers on Amazon

New (2) from ₹899.00 FREE Delivery.