ADSP 31012 IP01 Data Engineering Platforms for Analytics

- E-commerce Customer Behavior Analysis -

Yi Yao, Hira Stanley, Cassandra Maldonado, Apoorva Prakash, Kaiyan Li

1. Executive Summary

This project examines customer behavior within the electronics category on an e-commerce platform, leveraging over 516,000 records of interactions such as product views, additions to the cart, and purchases. By analyzing detailed activity data, including brand preferences, price sensitivity, and seasonal patterns, we aim to uncover the primary drivers of engagement and conversions for electronic products.

The analysis spans September 2020 to February 2021, a critical period for sales due to major shopping events like Black Friday, Cyber Monday, and pre-Christmas promotions. The findings will guide data-driven marketing strategies, improve inventory management, and enhance customer satisfaction, ultimately driving profitability for the electronics segment.

2. Business Case and Objectives

Business Case

The e-commerce platform aims to:

- Optimize customer engagement strategies tailored specifically to the electronics category.
- Identify trends in customer preferences across subcategories (e.g., computers, peripherals, network devices).
- Improve customer satisfaction by addressing pain points such as cart abandonment.
- Maximize profitability by aligning promotions and pricing strategies with customer buying behavior and seasonal patterns.

Objectives

- Understand customer behavior such as trends in views, cart additions, and purchases specific to electronics subcategories and brands.
- Assess the impact of pricing, brand loyalty, and promotions on purchase decisions within electronics.
- Explore shifts in customer behavior during major shopping events, such as Black Friday and Christmas, and identify peak sales periods for electronics.

3. Data Models

Conceptual Model

The conceptual model includes entities like Customers, Products, Categories, Events, and Transactions. Relationships are defined based on interactions within the e-commerce platform.

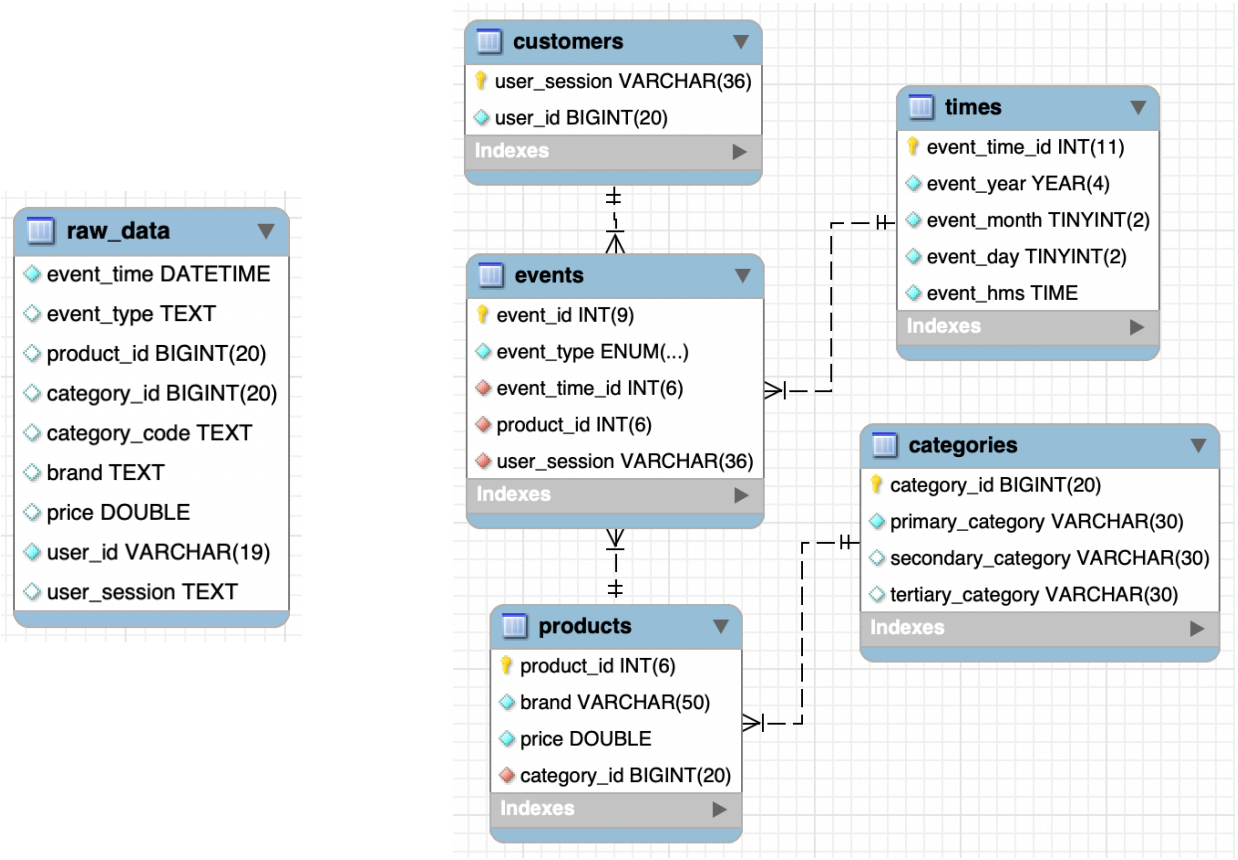
Entities and Relationships

The following are the key entities identified in our dataset, along with their relationships:

- **Categories:**
  - Represents the hierarchical structure of product categories.
  - Attributes: primary_category, secondary_category, tertiary_category.
  - Relationship: One-to-Many with Products.
- **Customers:**
  - Represents unique users identified by sessions.
  - Attributes: user_session, user_id.
  - Relationship: One-to-Many with Events.
- **Times:**
  - Represents the temporal aspects of events.
  - Attributes: event_year, event_month, event_day, event_hms.
  - Relationship: One-to-Many with Events.
- **Products:**
  - Represents products with associated details.
  - Attributes: product_id, brand, price, category_id.
  - Relationship: One-to-Many with Events and Many-to-One with Categories.

- **Events:**
  - Represents user interactions with products.
  - Attributes: event_type, event_time_id, user_session, product_id.
  - Relationship: Many-to-One with Times, Customers, and Products.

EER Diagram



Logical Model

Normalized to 3NF with entities for customer details, product categories, event types, and timestamps.

Schema Definition

Here is the logical schema for the normalized database:

1. **Categories:**
   - category_id (PK): BIGINT(20), Auto-Increment.

- primary_category: VARCHAR(30), Not Null.
- secondary_category: VARCHAR(30), Nullable.
- tertiary_category: VARCHAR(30), Nullable.

2. **Customers:**
   - user_session (PK): VARCHAR(36), Not Null.
   - user_id: BIGINT(20), Not Null.
3. **Times:**
   - event_time_id (PK): INT(11), Auto-Increment.
   - event_year: YEAR(4), Not Null.
   - event_month: TINYINT(2), Not Null.
   - event_day: TINYINT(2), Not Null.
   - event_hms: TIME, Not Null.
4. **Products:**
   - product_id (PK): INT(6), Not Null.
   - brand: VARCHAR(50), Not Null.
   - price: DOUBLE, Not Null.
   - category_id (FK): BIGINT(20), Not Null (references Categories.category_id).
5. **Events:**
   - event_id (PK): INT(9), Auto-Increment.
   - event_type: ENUM(view, cart, purchase), Not Null.
   - event_time_id (FK): INT(6), Not Null (references Times.event_time_id).
   - user_session (FK): VARCHAR(36), Not Null (references Customers.user_session).
   - product_id (FK): INT(6), Not Null (references Products.product_id).

Physical Model

Implementation Details

1. Database Management System: MySQL 8.0
2. Storage Engine: InnoDB
3. Collation: utf8mb4_0900_ai_ci for Unicode support.
4. Indexes:
   - Primary keys and foreign keys are indexed by default.

- Additional indexes added to frequently queried columns for performance.

DDL Statements

Creating Tables

```
CREATE DATABASE IF NOT EXISTS electronics;
USE electronics;
CREATE TABLE IF NOT EXISTS categories (
    category_id BIGINT(20) NOT NULL,
    primary_category VARCHAR(30) NOT NULL,
    secondary_category VARCHAR(30) DEFAULT NULL,
    tertiary_category VARCHAR(30) DEFAULT NULL,
    PRIMARY KEY (category_id)
) ENGINE=InnoDB;

CREATE TABLE IF NOT EXISTS customers (
    user_session VARCHAR(36) NOT NULL,
    user_id BIGINT(20) NOT NULL,
    PRIMARY KEY (user_session)
) ENGINE=InnoDB;

CREATE TABLE IF NOT EXISTS times (
    event_time_id INT NOT NULL AUTO_INCREMENT,
    event_time DATETIME NOT NULL,   # temporary variables, added for easier joins
    event_year YEAR(4) NOT NULL,
    event_month TINYINT(2) NOT NULL,
    event_day TINYINT(2) NOT NULL,
    event_hms TIME NOT NULL,
    PRIMARY KEY (event_time_id)
) ENGINE=InnoDB;

CREATE TABLE IF NOT EXISTS products (
```

```
    product_id INT(6) NOT NULL,
    brand VARCHAR(50) NOT NULL,
    price DOUBLE NOT NULL,
    category_id BIGINT(20) NOT NULL,
    PRIMARY KEY (product_id),
    FOREIGN KEY (category_id) REFERENCES categories (category_id)
) ENGINE=InnoDB;

CREATE TABLE IF NOT EXISTS events (
    event_id INT(9) NOT NULL AUTO_INCREMENT,
    event_type ENUM('view', 'cart', 'purchase') NOT NULL,
    event_time_id INT(6) NOT NULL,
    product_id INT(6) NOT NULL,
    user_session VARCHAR(36) NOT NULL,
    PRIMARY KEY (event_id),
    FOREIGN KEY (event_time_id) REFERENCES times (event_time_id),
    FOREIGN KEY (product_id) REFERENCES products (product_id),
    FOREIGN KEY (user_session) REFERENCES customers (user_session)
) ENGINE=InnoDB;
```

Normalization

Definitions

1. **First Normal Form (1NF)**

   A table is in 1NF if:

   - All columns contain atomic values (indivisible units of data).
   - Each column contains values of the same type.
   - The table has no repeating groups or arrays.

2. **Second Normal Form (2NF)**

   A table is in 2NF if:

   - It is already in 1NF.

- Every non-key column is fully functionally dependent on the entire primary key, not just a part of it.

3. Third Normal Form (3NF)

A table is in 3NF if:

- It is already in 2NF.
- There are no transitive dependencies (non-key columns depending on other non-key columns).

Steps Taken for Each Table

1. Times

- The raw_data table had a single column event_time, which contained full timestamp values (YYYY-MM-DD HH:MM:SS).
- To achieve 1NF, event_time should be split into separate columns: event_year, event_month, event_day, and event_hms. Each column now contains atomic values, ensuring uniform data types with no duplicates.
- To achieve 2NF, we need to make sure that attributes like event_year, event_month, event_day, and event_hms are derived directly from event_time_id and not from any subset of the composite key (since event_time_id is a single-column primary key).
- To achieve 3NF, the temporary event_time column, used for facilitating joins, must be removed as columns like event_year, event_month, and others are dependent on it. Once the joining process is complete, ensure this column is deleted.

```
INSERT INTO times (event_time, event_year, event_month, event_day, event_hms)
SELECT DISTINCT
   STR_TO_DATE(event_time, '%Y-%m-%d %H:%i:%s') AS event_time,
   YEAR(event_time) AS event_year,
   MONTH(event_time) AS event_month,
   DAY(event_time) AS event_day,
   TIME(event_time) AS event hms
FROM raw_data
```

```
ON DUPLICATE KEY UPDATE
    event_time = VALUES(event_time),
    event_year = VALUES(event_year),
    event_month = VALUES(event_month),
    event_day = VALUES(event_day),
    event_hms = VALUES(event_hms);

ALTER TABLE `times` DROP COLUMN `event_time`;  # After all inserting/joining processes
```

2. Categories

- The raw_data table contained a single column category_code, which combined multiple levels of category hierarchy (e.g., electronics.telephone).
- To achieve 1NF, category_code was split into primary_category, secondary_category, and tertiary_category, ensuring atomic values and uniform data types, with no repeating groups or arrays.
- To achieve 2NF, all non-key attributes (primary_category, secondary_category, tertiary_category) were fully dependent on the primary key category_id, ensuring no partial dependencies.
- To achieve 3NF, there are no transitive dependencies between the non-key attributes, as each column directly depends on category_id.

```
INSERT INTO categories (category_id, primary_category, secondary_category, tertiary_category)
    SELECT DISTINCT
        category_id,
        SUBSTRING_INDEX(category_code, '.', 1) AS primary_category,
        SUBSTRING_INDEX(SUBSTRING_INDEX(category_code, '.', 2), '.', -1) AS secondary_category,
        SUBSTRING_INDEX(category_code, '.', -1) AS tertiary_category
    FROM raw_data
```

```
ON DUPLICATE KEY UPDATE
    primary_category = VALUES(primary_category),
    secondary_category = VALUES(secondary_category),
    tertiary_category = VALUES(tertiary_category);
```

3. Products

- The raw_data table contained columns for product_id, brand, price, and category_id.
- To achieve 1NF, each product's attributes were stored in individual columns with atomic values, ensuring uniform data types and no duplicates.
- To achieve 2NF, non-key attributes (brand, price, category_id) were made fully dependent on the composite primary key (product_id).
- To achieve 3NF, category_id was normalized into the categories table, eliminating transitive dependencies, as category_id no longer stores hierarchical data directly in the products table.

```
INSERT INTO `products` (product_id, brand, price, category_id)
SELECT DISTINCT
    cb.product_id,
    cb.brand,
    cb.price,
    cb.category_id
FROM raw_data cb
ON DUPLICATE KEY UPDATE
    brand = VALUES(brand),
    price = VALUES(price),
    category_id = VALUES(category_id);
```

4. Customers

- The raw_data table included user_session and user_id as columns.
- Similarly, normalization can be done using the code below.

```
INSERT INTO `customers` (user_id, user_session)
SELECT DISTINCT user_id, user_session
FROM raw_data
ON DUPLICATE KEY UPDATE
   user_id = VALUES(user_id);
```

5. Events

- The raw_data table contained columns such as event_type, event_time, product_id, and user_session.
- Note: the event_time column was removed after normalization into the times table.
- Similarly, normalization can be done using the code below.

```
INSERT INTO `events` (event_type, event_time_id, product_id, user_session)
SELECT DISTINCT
   cb.event_type,
   et.event_time_id,
   cb.product_id,
   cb.user_session
FROM raw_data cb
JOIN times et
   ON cb.event_time = et.event_time
ON DUPLICATE KEY UPDATE
   event_type = VALUES(event_type),
   product_id = VALUES(product_id),
   user_session = VALUES(user_session);
```

4. Data Profiling

Key Observations and Cleaning Steps

1. Missing Values

- Observation:
  - Critical fields like product_id, user_session, and event_type contained missing values.
  - Non-critical fields such as category_code, user_id, and price also had missing entries.
- Impact:
  - Missing values in critical fields prevented certain rows from being included in the final tables.
  - Missing category_code and user_id led to incomplete parent table entries.
- Resolution:
  - For critical fields (product_id, user_session, event_type), we removed the rows with missing values.

2. Duplicate Entries

- Observation:
  - Duplicate user_session values caused primary key violations during insertion into the customers table.
  - The raw dataset contained duplicate rows across multiple fields, especially in product_id and event_time.
- Impact:
  - Duplicate user_session entries caused errors in child tables (events).
- Resolution:
  - We retained only the last occurrence of duplicate user_session values.

3. Invalid Timestamps

- Observation:
  - The event_time column contained timestamps in inconsistent formats (9/24/20 11:57 and 2020-09-24 11:57:00).
- Impact:
  - Parsing errors when inserting into the times table.
- Resolution:

- We standardized the event_time format using Python and removed UTC time.

4. Outliers in Numerical Data

- Observation:
  - The price column contained extreme values that skewed the dataset.
- Impact:
  - We created a visualization of the top product outliers.

5. Data Inconsistencies

- Observation:
  - category_code entries were inconsistent, containing multiple levels of categories combined into a single string (e.g., electronics.telephone).
- Impact:
  - Parsing category_code into separate fields (primary_category, secondary_category, tertiary_category) was challenging.
- Resolution:
  - Missing levels in category_code (e.g., secondary_category and tertiary_category) were defaulted to NULL, ensuring data integrity and maintaining a consistent hierarchy structure.

5. Methodology and Tools

ETL Process

1. Extract

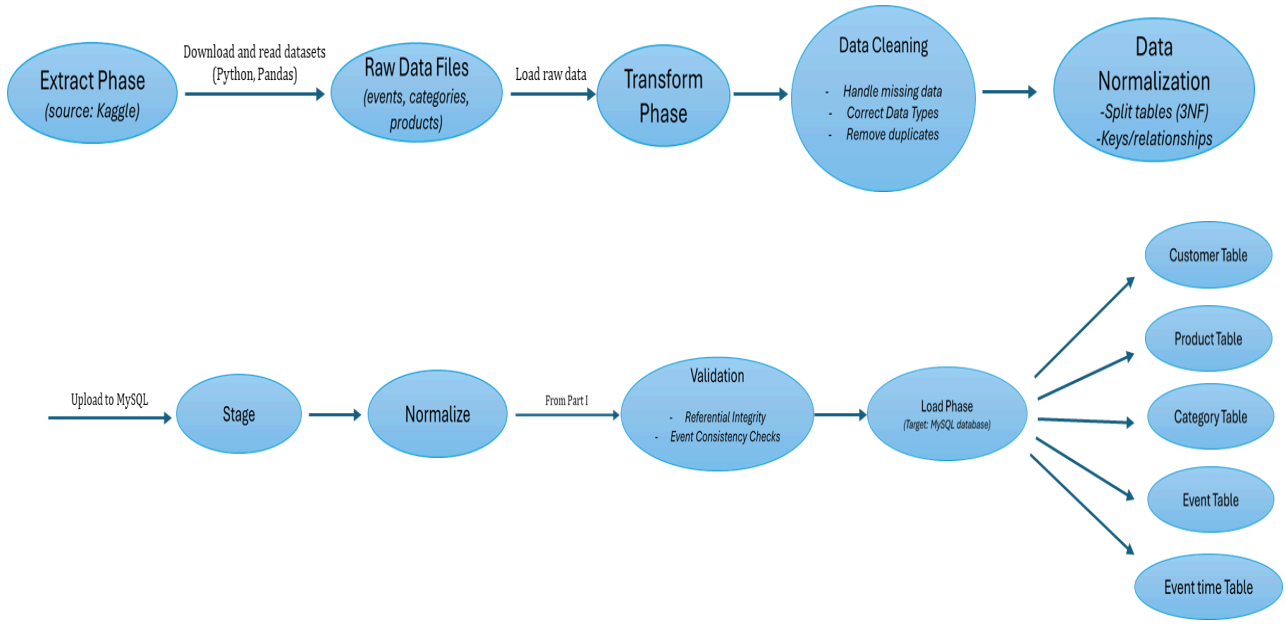
- Source: The raw_data table served as the source of truth.
- Data Format: Columns included event_time, product_id, user_session, category_code, etc.
- Tools: Data were extracted using Python and SQL.

2. Transform

- Key Transformations:
  - *Timestamp Normalization:* Extracted event_year, event_month, event_day, and event_hms from event_time for insertion into the times table.
  - *Category Hierarchy:* Processed category_code into primary_category, secondary_category, and tertiary_category.
  - *Deduplication:* Removed duplicate rows while ensuring the latest data was retained.
  - *Data Validation:* Checked for missing, invalid, or malformed data and applied corrections where necessary.
- Tools: Used Python for data manipulation and SQL for staging and validation.

3. Load

- Destination: Normalized tables (categories, customers, times, products, events).
- Process:
  - Inserted data into parent tables (categories, customers, times).
  - Populated child tables (products, events) with proper foreign key references.
- Tools: Used SQL queries to handle insertions, ensuring foreign key constraints and avoiding duplicates.



Tools and Technologies

- MySQL: Used for database management and querying.
- Python: Used for data cleaning, transformation, and visualization.
- Tableau: Used for creating reports and dashboards.

The process successfully transformed the raw dataset into a normalized relational schema, enabling better data management and analysis. Key achievements include:

- *Data Cleaning*: Removed missing, duplicate, and invalid data.
- *Normalization:* Established referential integrity across multiple tables.
- *Efficiency*: Optimized data for querying and analysis through deduplication and indexing.

Automation Methodology

End-to-End Pipeline Automation

To streamline the data processing workflow, an automated ETL pipeline is proposed:

1.Data Extraction: Use Python scripts with scheduled CRON jobs or Airflow to automate the extraction of raw data from the e-commerce database and other sources (for example: APIs for supplementary datasets).

2.Data Transformation: Automate transformations like timestamp normalization, category hierarchy parsing, and deduplication using Python libraries. Also incorporate data validation scripts to flag and resolve anomalies automatically.

3.Data Loading: Set up scheduled database updates using tools like Airflow or SQL scripts for automated ingestion into the normalized schema.

4.Dashboard Updates: Use Tableau's data extract automation or Python-based visualization libraries for automatic refreshes of reports and dashboards.

5.Monitoring and Alerts: Include logging and monitoring tools to track pipeline performance, with alert notifications for any failures.

6. Business Insights

Findings on Sales

1. By Category
    a. Highest: Computers $3,397,505
    b. Lowest: Medicine $288
2. By Month
    a. Highest: January $1,213,977
    b. Lowest: September $76,483
3. By Hour of Day
    a. Highest: 11 am $276,263
    b. Lowest: 1 am $38,853

Findings on Activity

We were able to view a customer's activity through the lens of an "event" occurring, such as viewing a product, adding it to cart, or purchasing the product. We broke each phase down through various dimensions, such as product type and what hour of the day the event occurred. The main insight this type of activity provides is a shopper's motivation to buy. Many viewers of

products are casually browsing, so they only constitute a “View”, whereas those who are highly motivated buyers “add to cart” and eventually “purchase”.

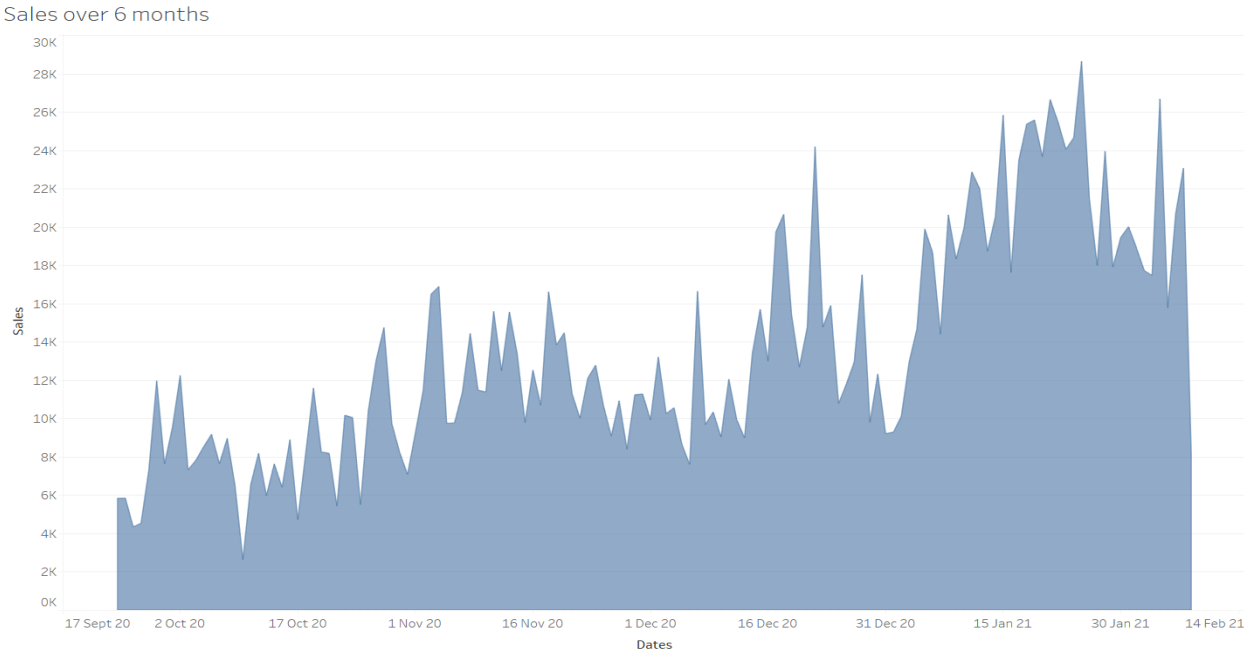
We did some feature engineering and created conversion rates based on the event types, specifically an “Add to Cart Rate” and a “Purchase Rate”. Add to Cart Rate achieved a maximum of about 10% for the computers category, and Purchase Rate achieved a maximum of about 90% for stationary. This tells us our real challenge is getting shoppers to click “Add to Cart”, which we can do through various marketing strategies.

Reports and Dashboards

Key Dashboards

Sales over time

We looked at data ranging from September 2020 to February 2021. We can see the seasonality of sales in this electronics ecommerce data set, as well as the peaks throughout the week and month. We wanted to investigate the event date/time dimension further by looking into Day of the week and other drilldowns on event time.

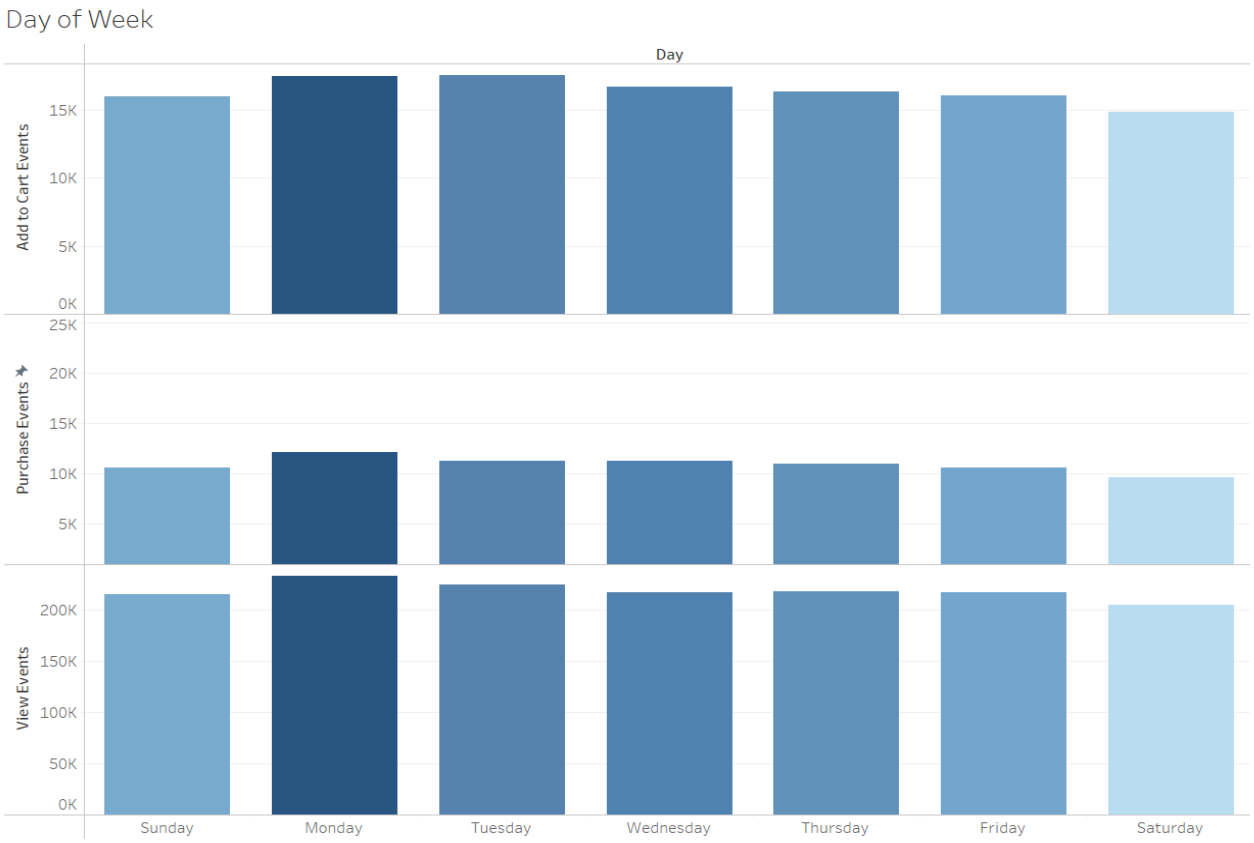


Activity by Day of Week

We see activity peaks on Mondays, with Tuesday and Wednesday following close behind. Most people tend to make purchases during the week, rather than the weekend, which was surprising

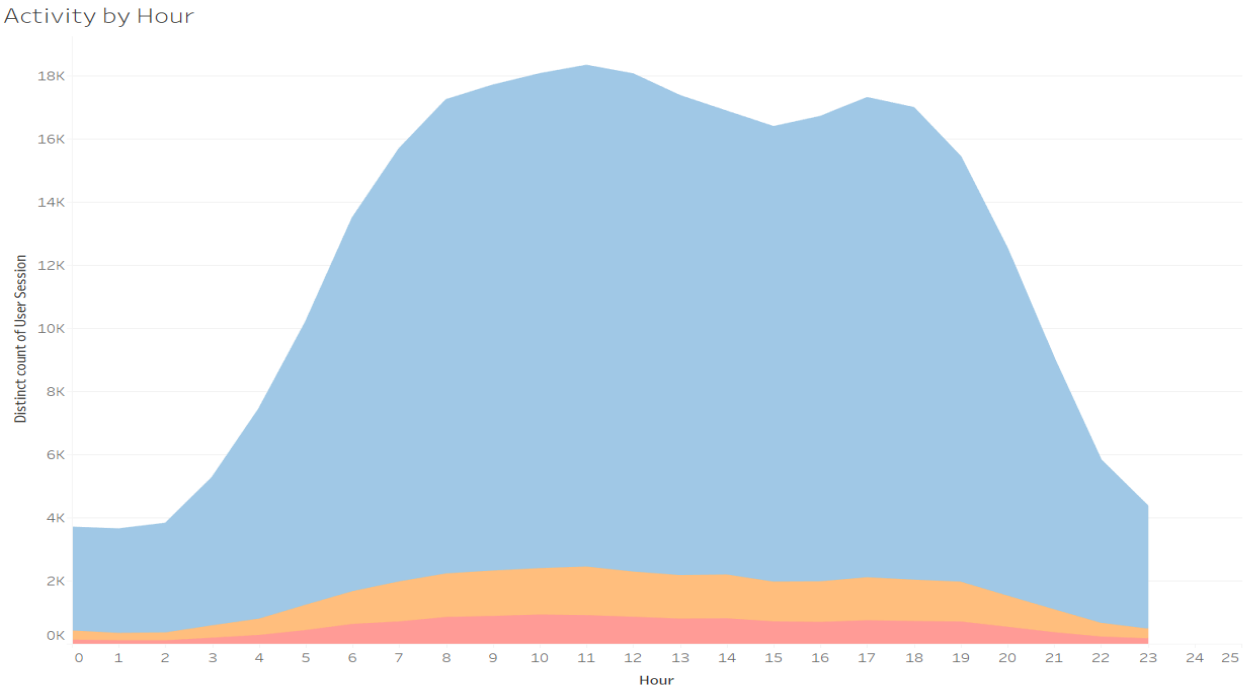
to us.

This can advise “flash sales” and promotions to capitalize on the higher traffic these days. We also know that demand is going to be higher on particular days so we can set up our inventory management system to accommodate for this so we don’t have a stockout and miss sales.



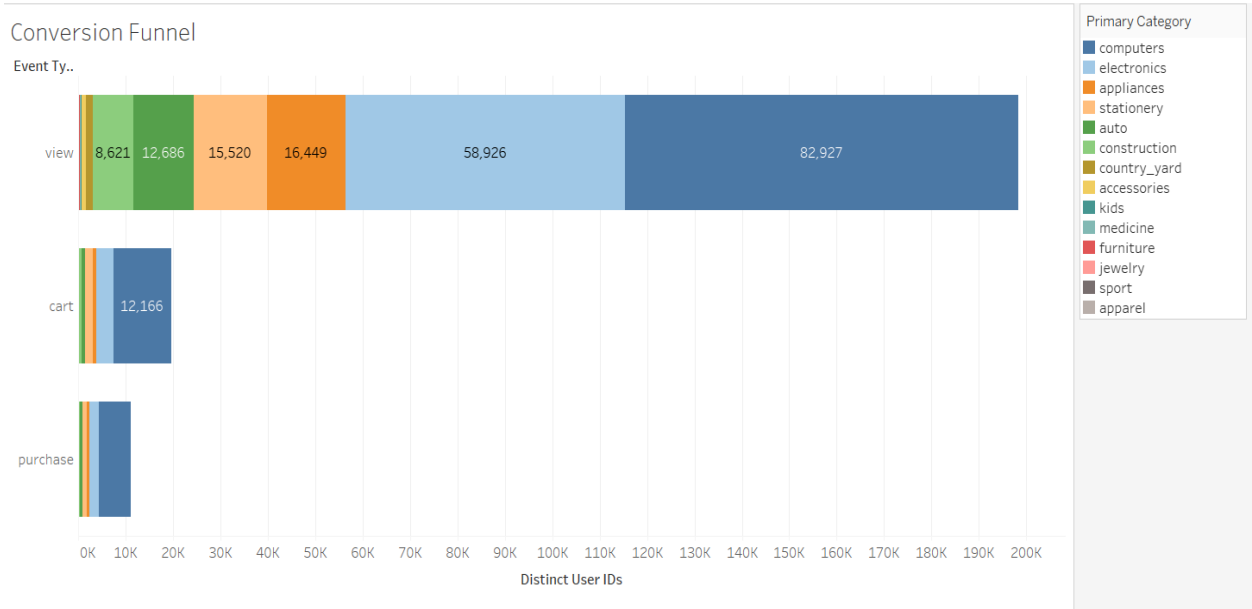
Activity by Hour fo the Day

If we drill down into hours of the day, we can see midmorning peak and post-work peak, with most people not really making any purchases through the night.



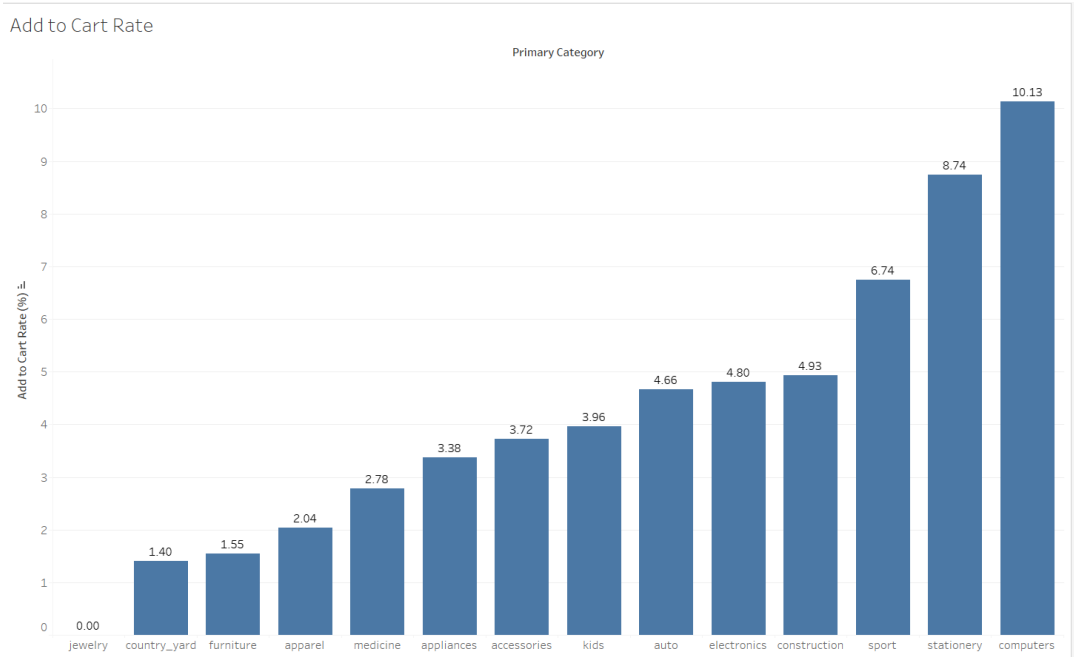
Conversion Funnel

This is what a typical eCommerce funnel looks like, with everyone viewing products, a portion of them adding those products to cart, and a portion of them purchasing the product. This displays a shopper's motivation to buy, between many viewers casually browsing vs. those who add it to their cart have strong intentions to buy. We broke it down by primary category to see the main drivers in each event, and can see computers and electronics driving this.



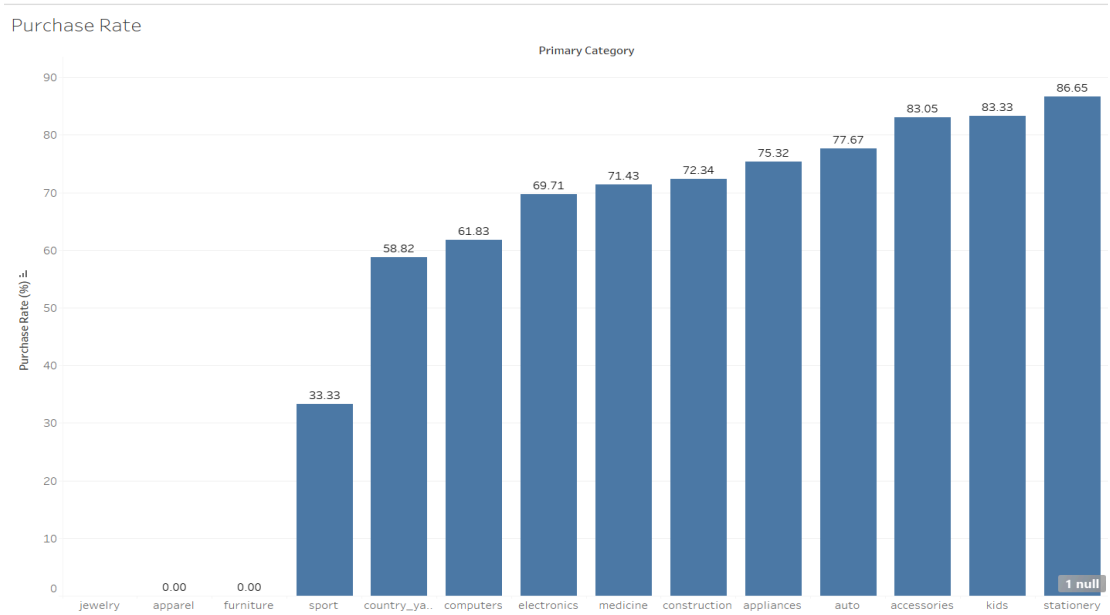
Add to Cart Rates by Category

We did some feature engineering, by turning those events into conversion rates. This one is what percentage of Views turned into Add to Cart actions.



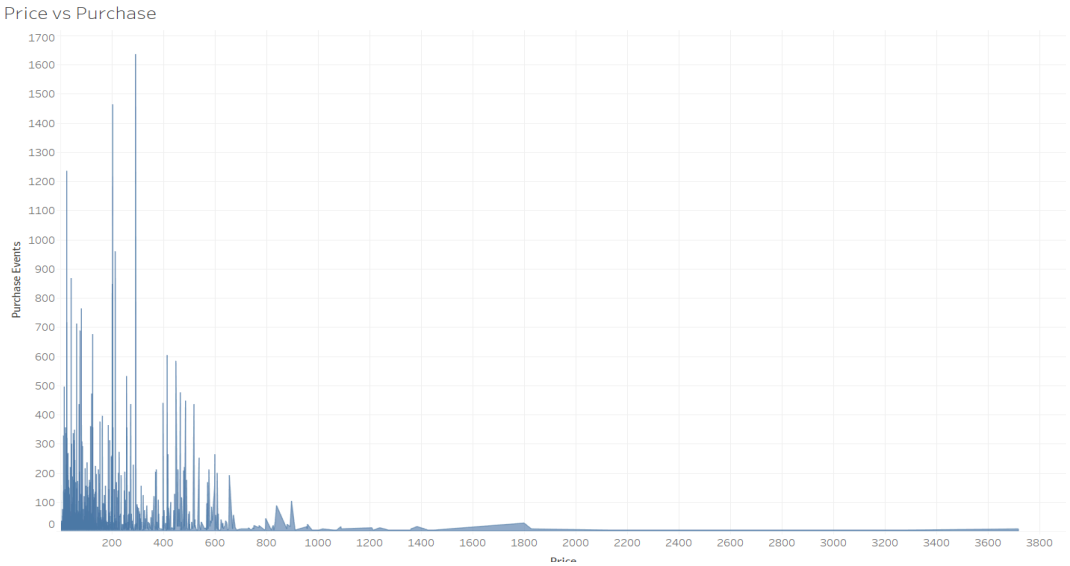
Purchase Rates by Category

Similarly, we did what percentage of the Add to Carts turned into actual Purchases. The alternative is they added to cart and then abandoned the cart.



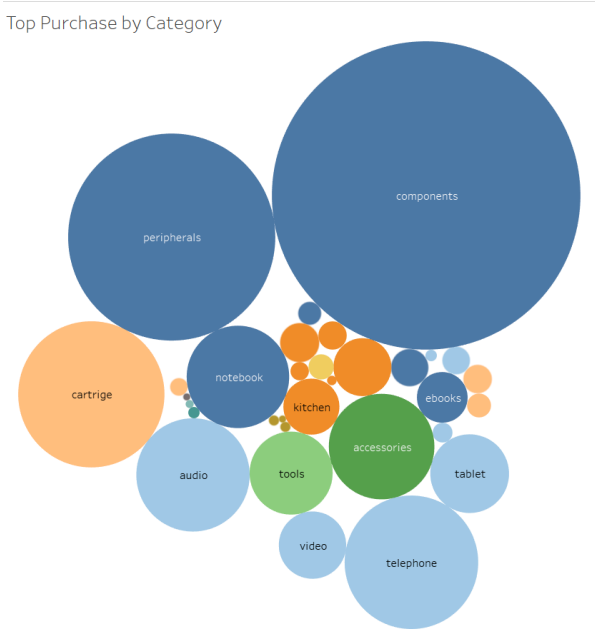
Price Sensitivity

Another insight we can see is high price sensitivity. Knowing what we know about computers and accessories being one of the biggest purchases, we can tell our shoppers are looking for more budget-friendly items, so we can position our products as “good value” or offer product lines specifically for lower-budget shoppers.

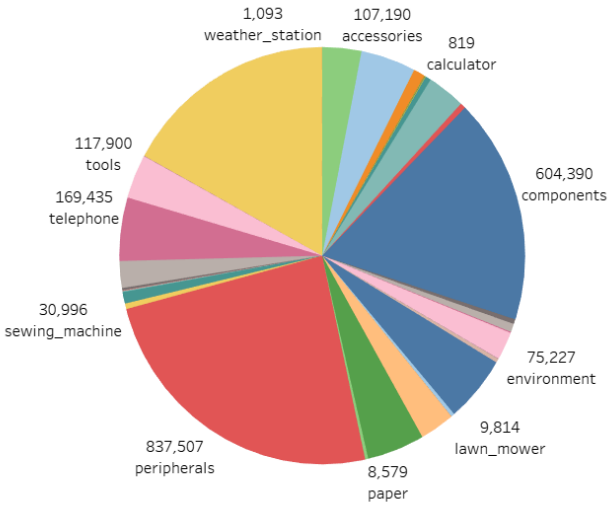


Top Purchases in Each Category

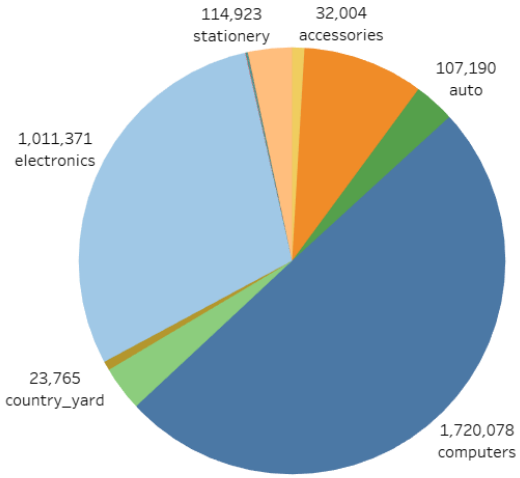
If we want to break down the previous visual more and see what was driving the computer and electronics categories, we can see here it’s very specific things. The biggest purchases are computer components and peripherals. This gives us insight into this eCommerce store’s customer base a little bit that they have strong affiliation with technology roles. A peripheral device is a hardware component that connects to a computer to enhance its functionality, but is not a core part of the computer.



Revenue by Primary Category



Revenue by Primary Category



7. Recommendations and Lessons Learned

<u>Recommendations</u>

- Marketing Strategies:

    - Focus on High-Conversion Categories: Prioritize marketing campaigns targeting categories with the highest purchase rates, such as mobile devices and accessories.
    - Capitalize on Holiday Periods: Implement promotions and flash sales during identified peak periods (Mondays and Wednesdays).
- Website Improvements:
    - Streamline Checkout Process: Simplify the checkout interface and integrate reminders for abandoned carts to minimize drop-off rates.
    - Optimize Server Performance: Ensure robust server capacity during peak hours to handle high traffic efficiently.
- Product Improvements
    - Introduce Product Bundles: Create cost-effective bundles to appeal to price-sensitive customers.
    - Improve Inventory Management: Avoid stockouts during high-demand periods by leveraging predictive analytics to forecast inventory needs.

Lessons Learned

1. Managing large datasets requires efficient ETL pipelines.
2. Data inconsistencies can be mitigated with robust cleaning methods.
3. Normalization aids in structured storage but may require denormalization for fast analysis.

8. References

Kaggle E-Commerce Data:
https://www.kaggle.com/datasets/mkechinov/ecommerce-purchase-history-from-electronics-store