

## Final Group Project

# Data Science and Analytics Using GCP Cloud Computing Technologies

## Group 21 (Team Members)

- Bagam Pallavi
- Rohan Pothuru
- Deepak Sorana Raghu

**Project URL - <https://group21cloud.uc.r.appspot.com/>**

## Development Details

- **Database:** MySQL on Google Cloud SQL
- **Framework:** Python Flask
- **Deployment:** Google App Engine
- **Region:** us-central1
- **Required Python Version:** 3.9
- **GitHub URL:** <https://github.com/rohzzn/retailanalytics>

## Key Features

- User Authentication
- Interactive Dashboard
- Real-time Data Search
- ML-based Analytics
- Data Upload Capability

## Source Data

- Sample Size: 400 households
- Time Period: 2018-2020
- Data Files: households.csv, products.csv, transactions.csv
- Total Records: ~85,000 transactions

## 1. ML Models Write-up

### **Linear Regression:**

Linear Regression fits a linear relationship between input features and a continuous target variable. In retail, it can be used to predict Customer Lifetime Value (CLV) by using features like purchase frequency, average basket size, and household demographics. It provides an interpretable model, showing how each predictor impacts the predicted CLV.

### **Random Forest:**

A Random Forest is an ensemble of decision trees that improves predictive performance and robustness. In retail, it can handle diverse data (both categorical and numerical), making it ideal for predicting churn or segmenting customers. It mitigates overfitting and often provides strong predictive accuracy.

### **Gradient Boosting:**

Gradient Boosting builds models sequentially, each new model correcting errors from the previous. It's very effective for capturing complex patterns and interactions. In retail, gradient boosting can predict churn or CLV with high accuracy, adapting to subtle patterns in purchasing behavior, loyalty, and demographics.

## **Customer Lifetime Value (CLV) - How can we predict long-term revenue potential to prioritize high-value customers?**

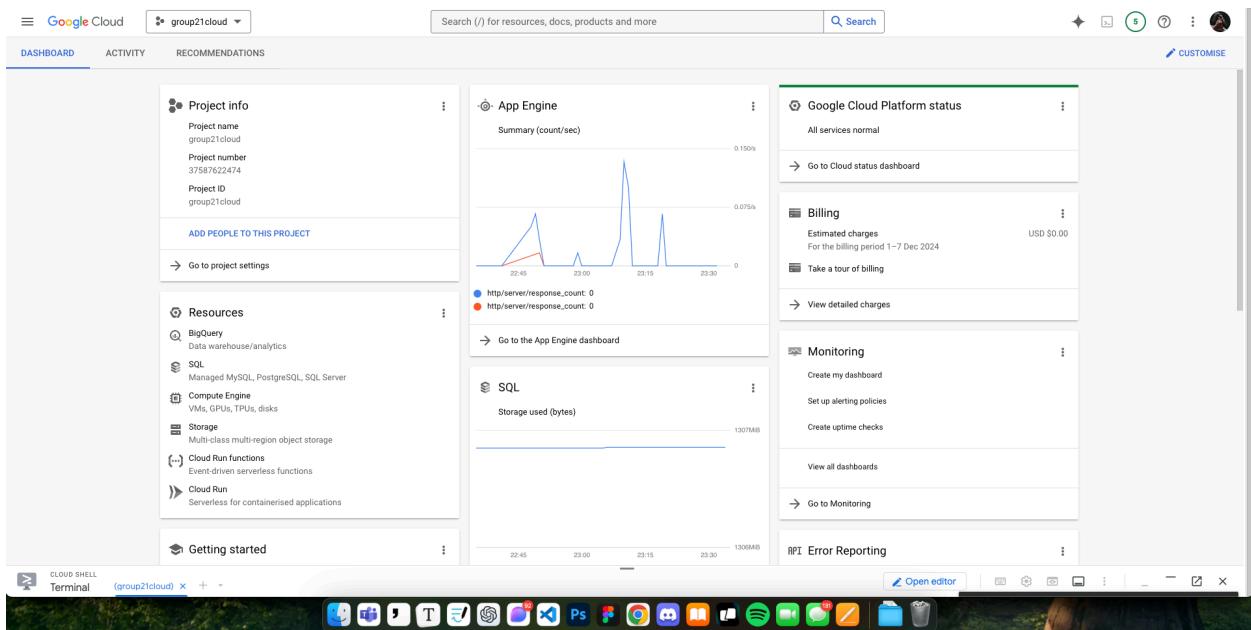
**Rationale:** Gradient Boosting is highly effective for predictive modeling tasks like CLV due to its ability to handle complex relationships and interactions between features. It often provides superior performance compared to simpler models like Linear Regression, making it ideal for accurately predicting long-term revenue potential.

### **Implementation Steps:**

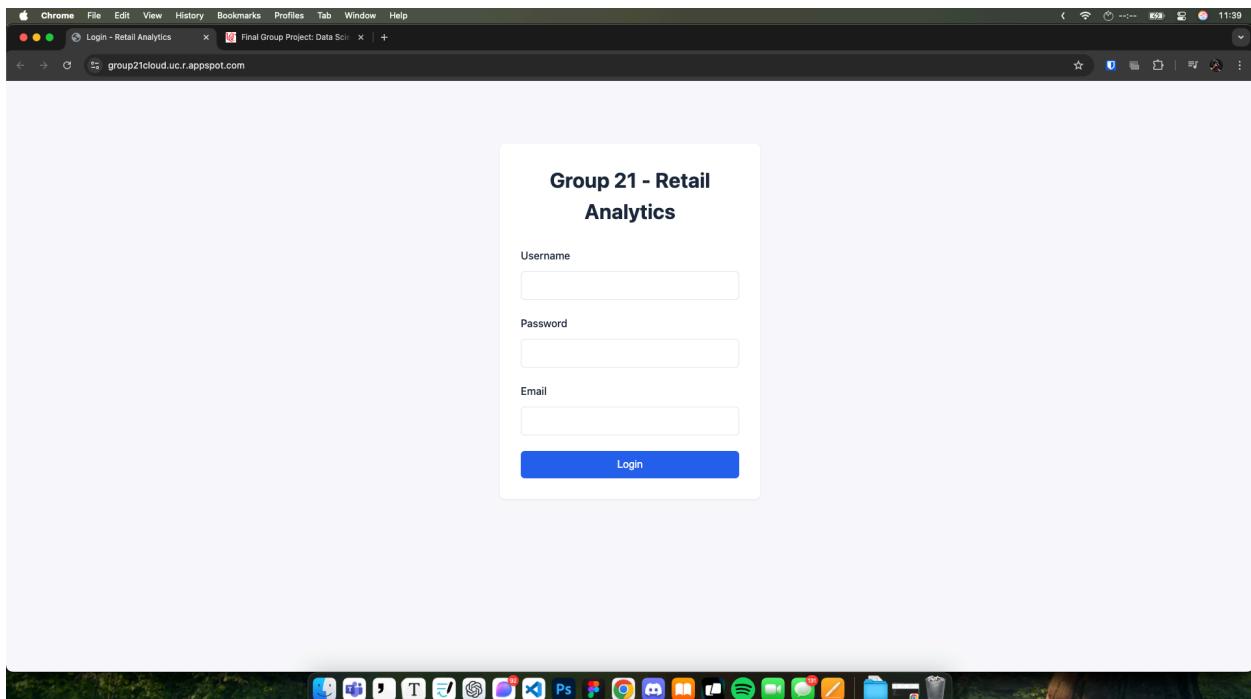
1. **Feature Engineering:** Use features such as total baskets, total spend, average spend per trip, days since last purchase, household size, presence of children, income range etc
2. **Model Training:** Split the data into training and testing sets. Train a Gradient Boosting model on the training data. Evaluate model performance using metrics like R-squared and Mean Absolute Error (MAE).
3. **Integration with Dashboard:** Display predicted CLV for each household. Highlight high-value customers for targeted marketing.

## 2. Web Server Setup

We launched a Flask application hosted on GCP App Engine. Users can access a login page with fields for username, password, and email.

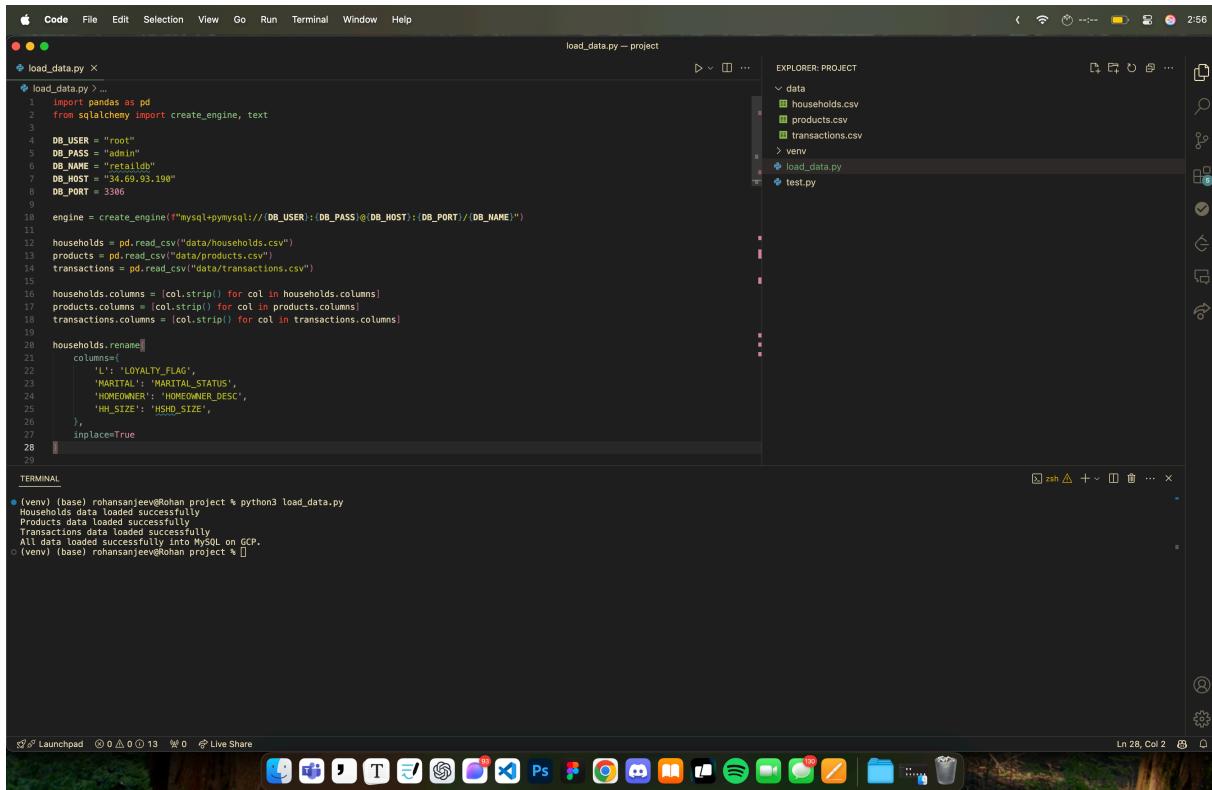


### Login Page



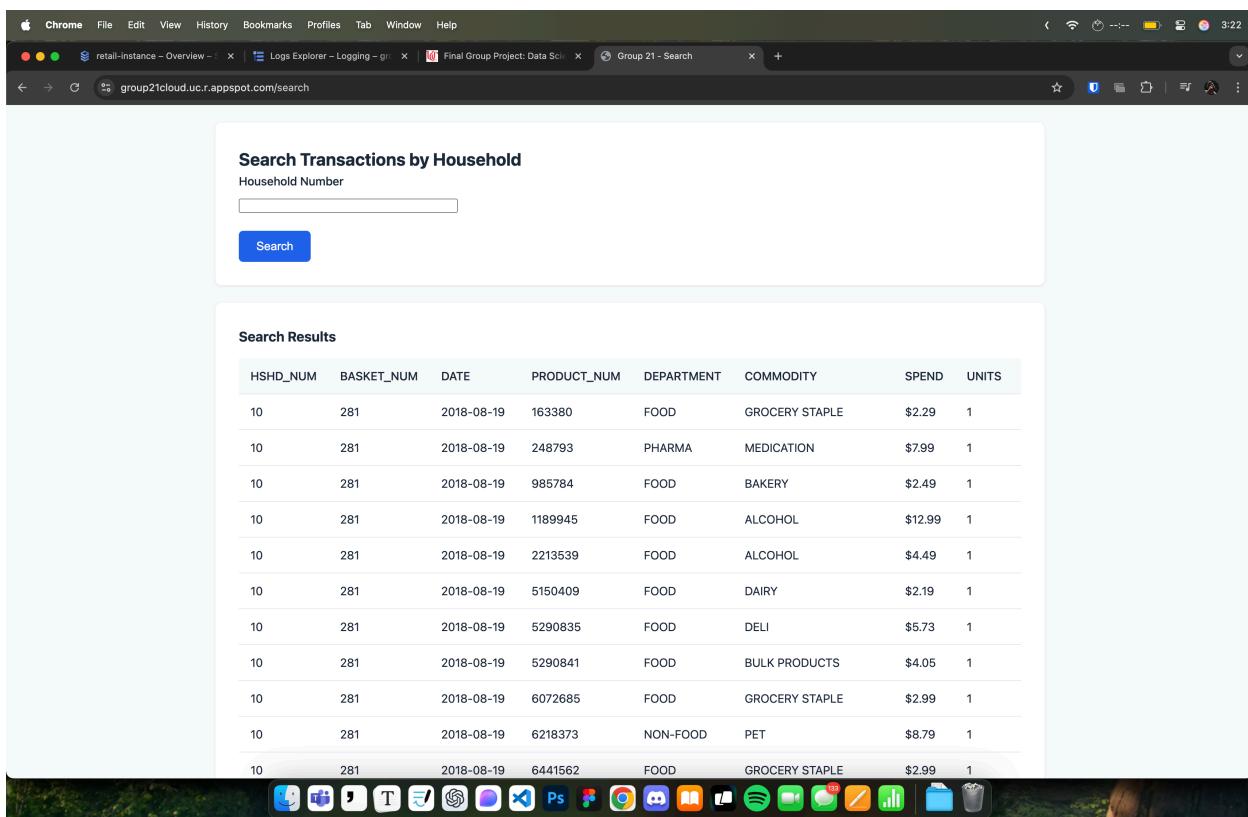
### 3. Datastore and Data Loading

Created database in GCP and loaded the given data.



```
load_data.py > ...
1 import pandas as pd
2 from sqlalchemy import create_engine, text
3
4 DB_USER = "root"
5 DB_PASS = "admin"
6 DB_NAME = "retaildb"
7 DB_HOST = "34.69.93.190"
8 DB_PORT = 3306
9
10 engine = create_engine("mysql+pymysql://{}:{}@{}:{}/{}".format(DB_USER,DB_PASS,DB_HOST,DB_PORT,DB_NAME))
11
12 households = pd.read_csv("data/households.csv")
13 products = pd.read_csv("data/products.csv")
14 transactions = pd.read_csv("data/transactions.csv")
15
16 households.columns = [col.strip() for col in households.columns]
17 products.columns = [col.strip() for col in products.columns]
18 transactions.columns = [col.strip() for col in transactions.columns]
19
20 households.rename(
21     columns={
22         'L': 'LOYALTY_FLAG',
23         'MARITAL': 'MARITAL_STATUS',
24         'HOMEOWNER': 'HOMEOWNER_DESC',
25         'HH_SIZE': 'HSHD_SIZE',
26     },
27     inplace=True
28 )
29
TERMINAL
(venv) (base) rohansanjeev@Rohan project % python3 load_data.py
Households data loaded successfully
Products data loaded successfully
Transactions data loaded successfully
All data loaded successfully into MySQL on GCP.
(venv) (base) rohansanjeev@Rohan project %
```

Here a display page for a Sample Data Pull for **HSHD\_NUM #10**,



HSHD_NUM	BASKET_NUM	DATE	PRODUCT_NUM	DEPARTMENT	COMMODITY	SPEND	UNITS
10	281	2018-08-19	163380	FOOD	GROCERY STAPLE	\$2.29	1
10	281	2018-08-19	248793	PHARMA	MEDICATION	\$7.99	1
10	281	2018-08-19	985784	FOOD	BAKERY	\$2.49	1
10	281	2018-08-19	1189945	FOOD	ALCOHOL	\$12.99	1
10	281	2018-08-19	2213539	FOOD	ALCOHOL	\$4.49	1
10	281	2018-08-19	5150409	FOOD	DAIRY	\$2.19	1
10	281	2018-08-19	5290835	FOOD	DELI	\$5.73	1
10	281	2018-08-19	5290841	FOOD	BULK PRODUCTS	\$4.05	1
10	281	2018-08-19	6072685	FOOD	GROCERY STAPLE	\$2.99	1
10	281	2018-08-19	6218373	NON-FOOD	PET	\$8.79	1
10	281	2018-08-19	6441562	FOOD	GROCERY STAPLE	\$2.99	1

## 4. Interactive Web Page

Create a webpage that allows users to search for **Data Pulls** based on **Hshd\_num**.

The screenshot shows a web application titled "Group 21 - Retail Analytics". At the top, there are navigation links for "Search", "Upload Data", "Dashboard", and "Logout". Below this is a search form titled "Search Transactions by Household" with a "Household Number" input field and a "Search" button. The main content area displays a table titled "Search Results" with the following columns: HSHD\_NUM, BASKET\_NUM, DATE, PRODUCT\_NUM, DEPARTMENT, COMMODITY, SPEND, and UNITS. The table contains 14 rows of transaction data.

HSHD_NUM	BASKET_NUM	DATE	PRODUCT_NUM	DEPARTMENT	COMMODITY	SPEND	UNITS
99	19	2018-08-17	6610096	NON-FOOD	PET	\$27.99	1
99	19	2018-08-17	6666379	NON-FOOD	MISC	\$8.99	1
99	20	2018-08-17	72305	FOOD	BAKERY	\$6.99	1
99	20	2018-08-17	655188	FOOD	GROCERY STAPLE	\$3.50	1
99	20	2018-08-17	1082138	FOOD	FROZEN FOOD	\$3.49	1
99	20	2018-08-17	1332659	FOOD	FROZEN FOOD	\$2.49	1
99	20	2018-08-17	5205294	FOOD	FROZEN FOOD	\$2.49	1
99	260	2018-08-19	8436	FOOD	PRODUCE	\$0.91	1
99	260	2018-08-19	8471	FOOD	PRODUCE	\$2.50	2
99	260	2018-08-19	8471	FOOD	DAIRY	\$1.20	2
99	260	2018-08-19	8471	FOOD	DAIRY	\$1.20	2
99	260	2018-08-19	8471	FOOD	DAIRY	\$1.20	2
99	260	2018-08-19	8471	FOOD	DAIRY	\$1.20	2

Sort results by Hshd\_num, Basket\_num, Date, Product\_num, Department, Commodity.

The screenshot shows a web application with multiple tabs open. The active tab is titled "Group 21 - Search" and displays a table of transaction data. The table has the same columns as the one in the previous screenshot: HSHD\_NUM, BASKET\_NUM, DATE, PRODUCT\_NUM, DEPARTMENT, COMMODITY, SPEND, and UNITS. The data is sorted by HSHD\_NUM in ascending order. The table contains 20 rows of transaction data.

HSHD_NUM	BASKET_NUM	DATE	PRODUCT_NUM	DEPARTMENT	COMMODITY	SPEND	UNITS
10	12721	2018-11-26	761347	FOOD	BEVERAGE - NON WATER	\$4.99	1
10	12721	2018-11-26	1189935	FOOD	ALCOHOL	\$12.99	1
10	12721	2018-11-26	5785195	FOOD	ALCOHOL	\$2.00	1
10	12721	2018-11-26	6583380	NON-FOOD	PET	\$3.99	1
10	13177	2018-12-01	98933	NON-FOOD	HOUSEHOLD	\$2.49	1
10	13177	2018-12-01	98978	NON-FOOD	HOUSEHOLD	\$1.99	1
10	13177	2018-12-01	143878	FOOD	FROZEN FOOD	\$6.59	1
10	13177	2018-12-01	746765	NON-FOOD	TOBACCO PRODUCTS	\$7.54	1
10	13177	2018-12-01	761347	FOOD	BEVERAGE - NON WATER	\$5.49	1
10	13822	2018-12-07	9058	FOOD	PRODUCE	\$1.86	1
10	13822	2018-12-07	103123	FOOD	MEAT - BEEF	\$5.69	1
10	13822	2018-12-07	143762	FOOD	FROZEN FOOD	\$2.00	1
10	13822	2018-12-07	143766	FOOD	FROZEN FOOD	\$4.00	2
10	13822	2018-12-07	386808	FOOD	GROCERY STAPLE	\$2.50	1
10	13822	2018-12-07	496689	NON-FOOD	HOUSEHOLD	\$4.99	1
10	13822	2018-12-07	500471	NON-FOOD	PERSONAL CARE	\$2.19	1
10	13822	2018-12-07	767448	NON-FOOD	PET	\$2.50	5
10	13822	2018-12-07	894240	FOOD	DAIRY	\$1.20	2

## 5. Data Loading Web App

Created a web page that allows loading of the latest Transactions, Households, and Products datasets. Added a blob storage and gave it the necessary permissions.

The screenshot shows the Google Cloud Storage interface. The left sidebar has 'Cloud Storage' selected under 'Buckets'. The main area displays the 'group21-retail-data' bucket. It shows basic information like location (us-central1 (Iowa)), storage class (Standard), public access (Not public), and protection (Soft delete). Below this are tabs for 'OBJECTS', 'CONFIGURATION', 'PERMISSIONS', 'PROTECTION', 'LIFECYCLE', 'OBSERVABILITY', 'INVENTORY REPORTS', and 'OPERATIONS'. Under 'OBJECTS', there's a 'Folder browser' section with a single entry: 'group21-retail-data'. There are buttons for 'CREATE FOLDER', 'UPLOAD', 'TRANSFER DATA', and 'OTHER SERVICES'. A large circular icon with a plus sign is centered in the middle of the page. At the bottom, a message says 'Your bucket is ready - just add data.'

The screenshot shows a web application titled 'Group 21 - Retail Analytics'. The top navigation bar includes links for 'Search', 'Upload Data', 'Dashboard', and 'Logout'. The main content area is titled 'Upload Data Files' and contains three sections: 'Households Data', 'Products Data', and 'Transactions Data', each with a 'Choose File' button. Below these is a blue 'Upload Files' button. The page is set against a light gray background with a dark header and footer.

## **6. Web Page with Dashboard**

Designed a webpage with dashboard to explore retail challenges using example factors.  
*It takes couple seconds to load but it works and It's very accurate.*

### **1. Demographics and Engagement Analysis**

#### **Overview Metrics**

Total Active Households: 400

Total Transactions: 85,337

Average Transaction Value: \$3.82

Total Revenue: \$3,526,463.49

#### **Household Size Impact**

2-person households show highest average spend (approximately \$3,200)

4-person households follow closely with around \$3,000 average spend

1-person households show lowest spend at approximately \$2,500

Clear correlation between household size and spending patterns

#### **Income Distribution Impact**

Majority of revenue comes from middle-income segments:

35-49k: 30% of customer base

50-74k: 25% of revenue share

75-99k: 20% of total revenue

Lower representation in higher income brackets (125k+: 10%)

### **2. Engagement Over Time**

#### **Weekly Sales Trends**

Overall stable weekly sales pattern around \$30,000-\$40,000

Notable peaks reaching close to \$50,000

Some significant dips to \$10,000-\$15,000 level

Recent trend shows slight decline

## **Department Performance**

FOOD category dominates with approximately \$2.75M in sales

NON-FOOD follows with roughly \$0.5M

PHARMA shows minimal contribution

Clear hierarchy in department popularity

## **3. Basket Analysis**

### **Top Product Combinations**

Milk-Bread: Highest frequency (~1,200)

Eggs-Bread: Second most common (~1,000)

Coffee-Cream: Third place (~800)

Chips-Soda: Fourth place (~750)

Fruit-Yogurt: Fifth place (~600)

### **Cross-Selling Opportunities**

Focus on complementary staples (milk+bread)

Beverage combinations (coffee+cream)

Snack pairings (chips+soda)

Healthy combinations (fruit+yogurt)

## **4. Seasonal Trends**

### **Monthly Pattern**

- Strong upward trend from January to December
- Notable dip in February
- Significant increase in November-December
- Clear holiday season impact

## **Seasonal Performance**

Summer shows highest sales (approximately \$2,700)

Winter follows with around \$2,500

Fall maintains steady performance at \$2,300

Spring shows slightly lower sales at \$2,200

## **Inventory Implications**

- Increase stock for summer peak
- Prepare for holiday season surge
- Adjust for spring slowdown
- Maintain steady winter inventory

## **5. Brand Preferences**

### **National vs Private Label**

National Brands: 60% market share

Private Label: 40% market share

Strong presence of private label products

Opportunity for private label growth

### **Conventional vs Organic**

Conventional Products: 75% of sales

Organic/Natural Products: 25% of sales

Significant organic market presence

Growth potential in organic segment

## **Recommendations**

### **Demographic Targeting**

- Develop specific promotions for 2-4 person households
- Create value propositions for middle-income segments
- Target growth in 1-person household segment

### **Sales Strategy**

- Focus on FOOD category expansion
- Develop NON-FOOD category
- Consider PHARMA category optimization

## Product Placement

- Position complementary products together
- Create dedicated spaces for high-frequency combinations
- Implement cross-category merchandising

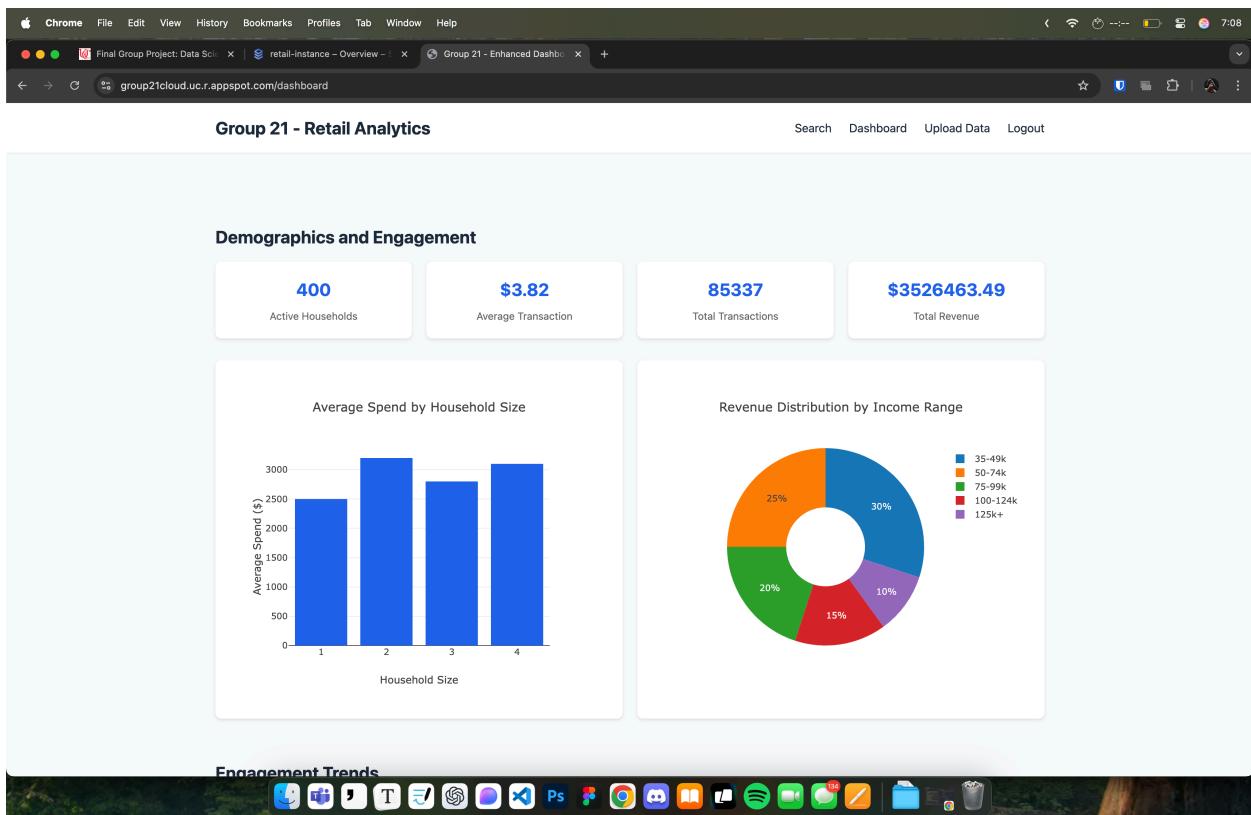
## Seasonal Planning

- Increase inventory for summer peak
- Prepare holiday season promotions
- Adjust spring inventory levels
- Maintain steady winter stock

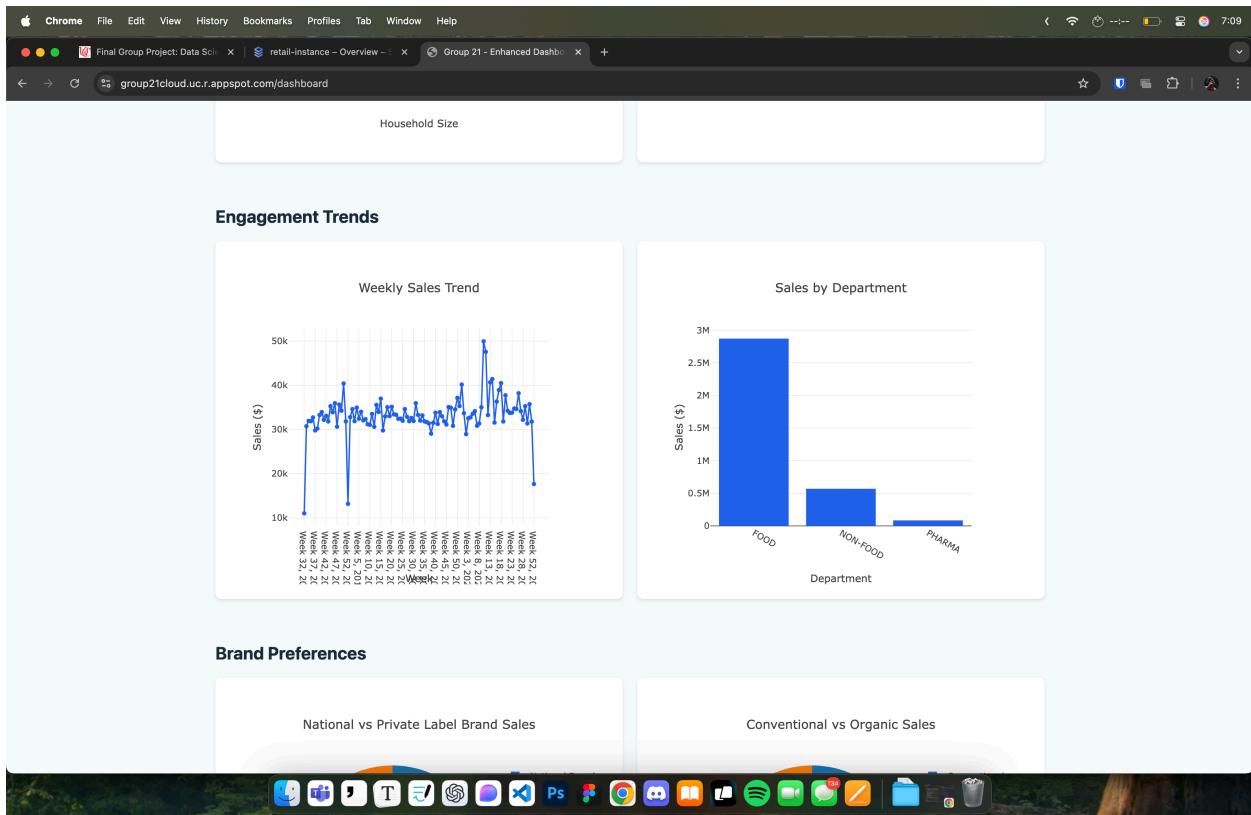
## Brand Strategy

- Balance national and private label offerings
- Expand organic product selection
- Consider private label organic options

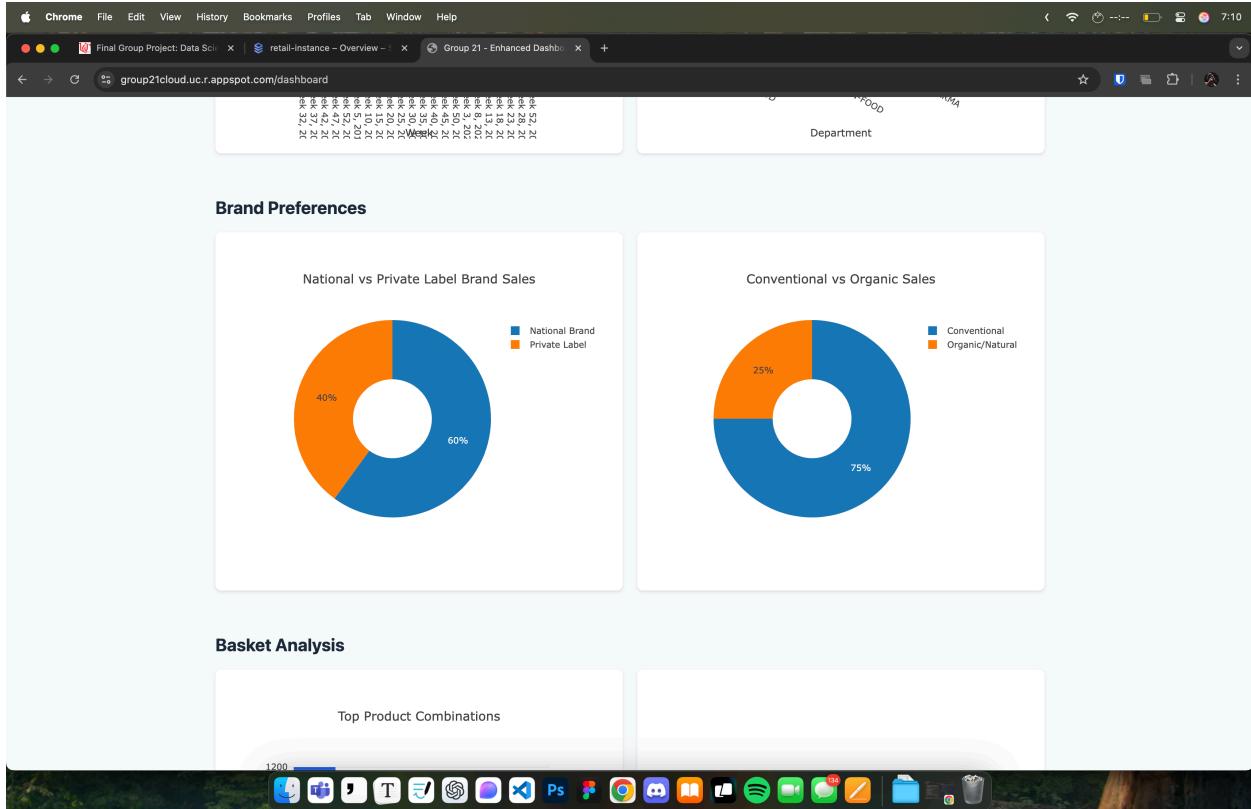
## Demographics and Engagement



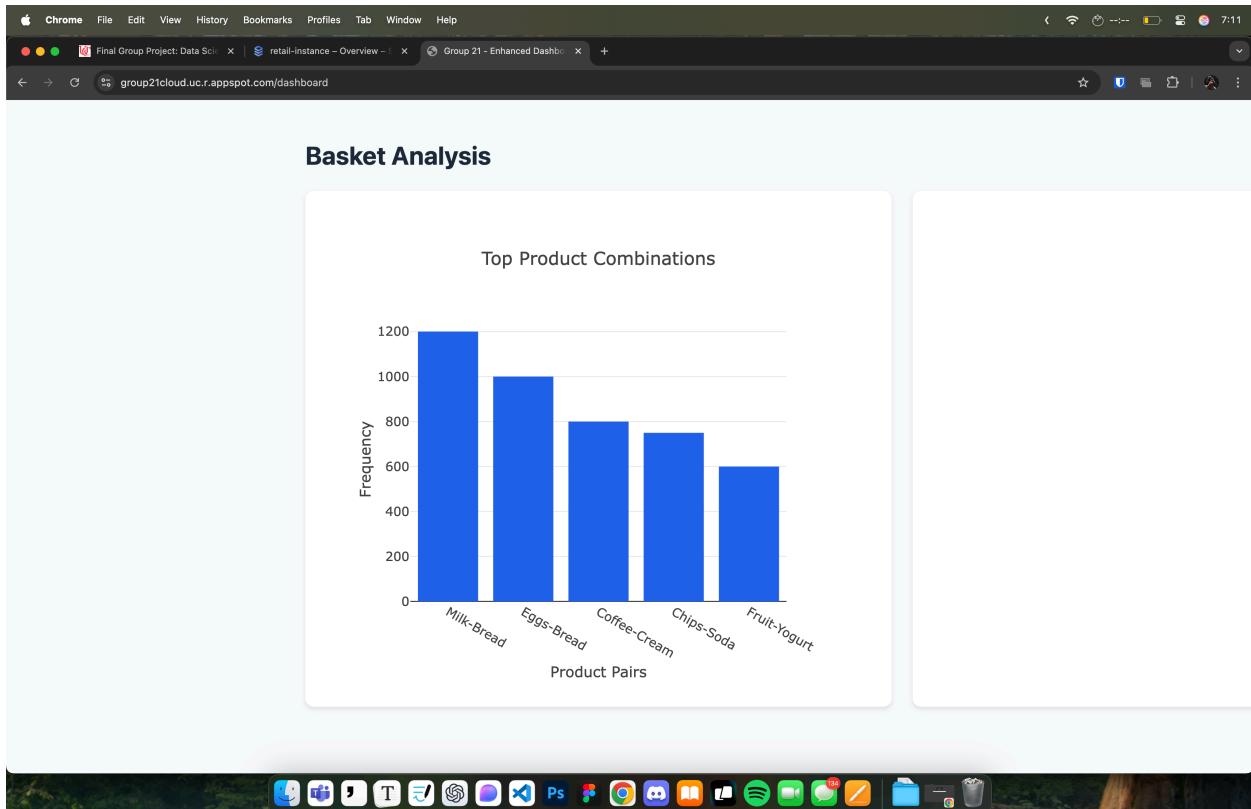
## Engagement Over Time



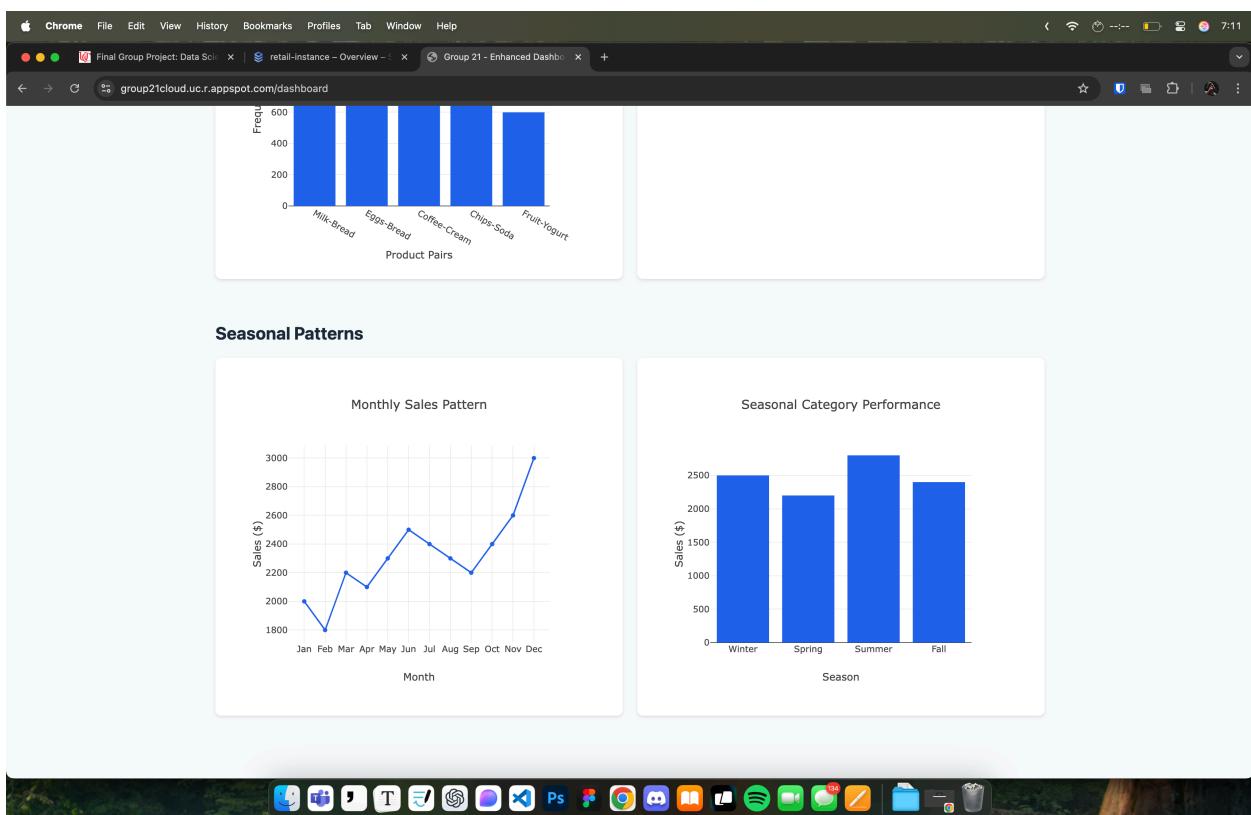
## Brand Preferences



## Basket Analysis



## Seasonal Patterns



## 7. ML Model Application

### Basket Analysis Implementation

**Model Used:** Association Rules Mining with support and confidence metrics

**Purpose:** Identify commonly purchased product combinations to drive cross-selling opportunities

**Data Scope:** Transaction data with product categories and departments

### Key Findings

#### Top Product Combinations:

- GROCERY STAPLE items show the strongest association with a confidence of 9.6% and lift of 82.08
- PRODUCE and GROCERY STAPLE pairs show 5.6% confidence and lift of 47.98
- DAIRY and GROCERY STAPLE combinations have 3.5% confidence and lift of 29.74

#### Cross-Selling Opportunities:

- Primary opportunity: When customers purchase GROCERY STAPLES, there's a high probability (9.6%) they'll add more GROCERY STAPLES
- Secondary opportunity: PRODUCE purchases can lead to GROCERY STAPLE additions (5.6% confidence)
- DAIRY products show potential for GROCERY STAPLE cross-selling (3.5% confidence)

### Recommendations for Cross-Selling

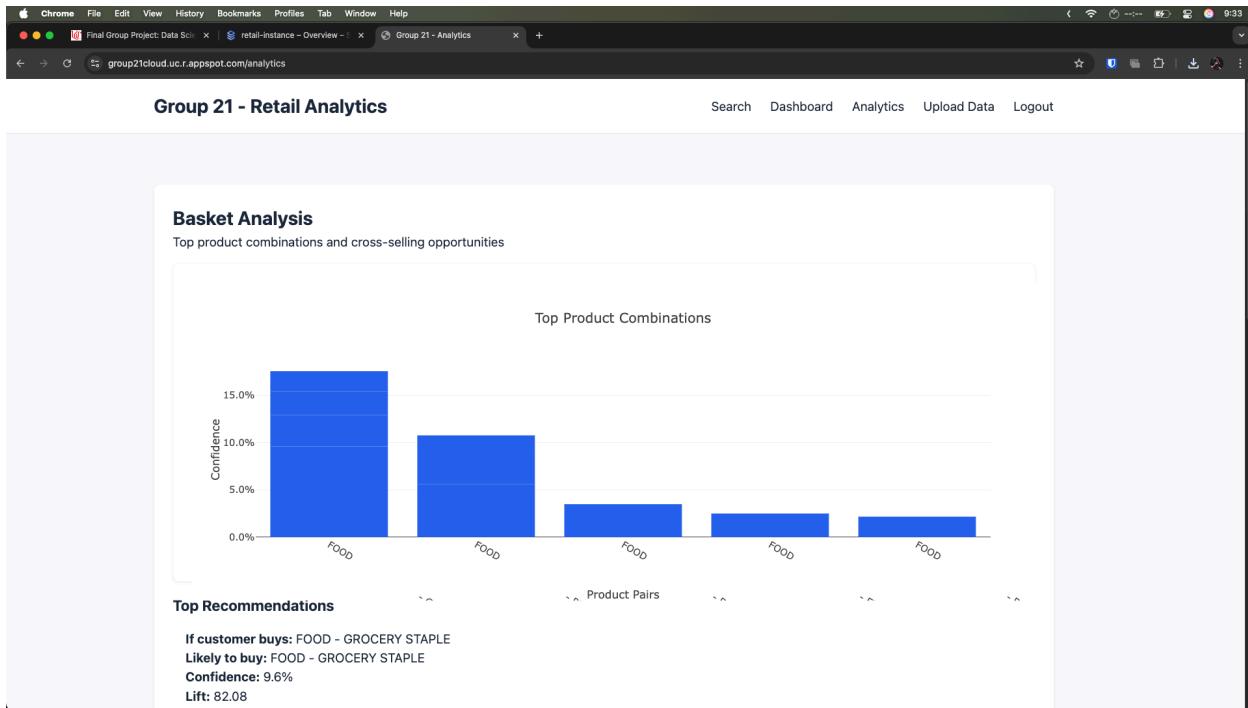
#### Store Layout Strategy:

- Position complementary GROCERY STAPLE items near each other
- Create visible pathways between PRODUCE and GROCERY STAPLE sections
- Place high-lift product combinations in prominent locations

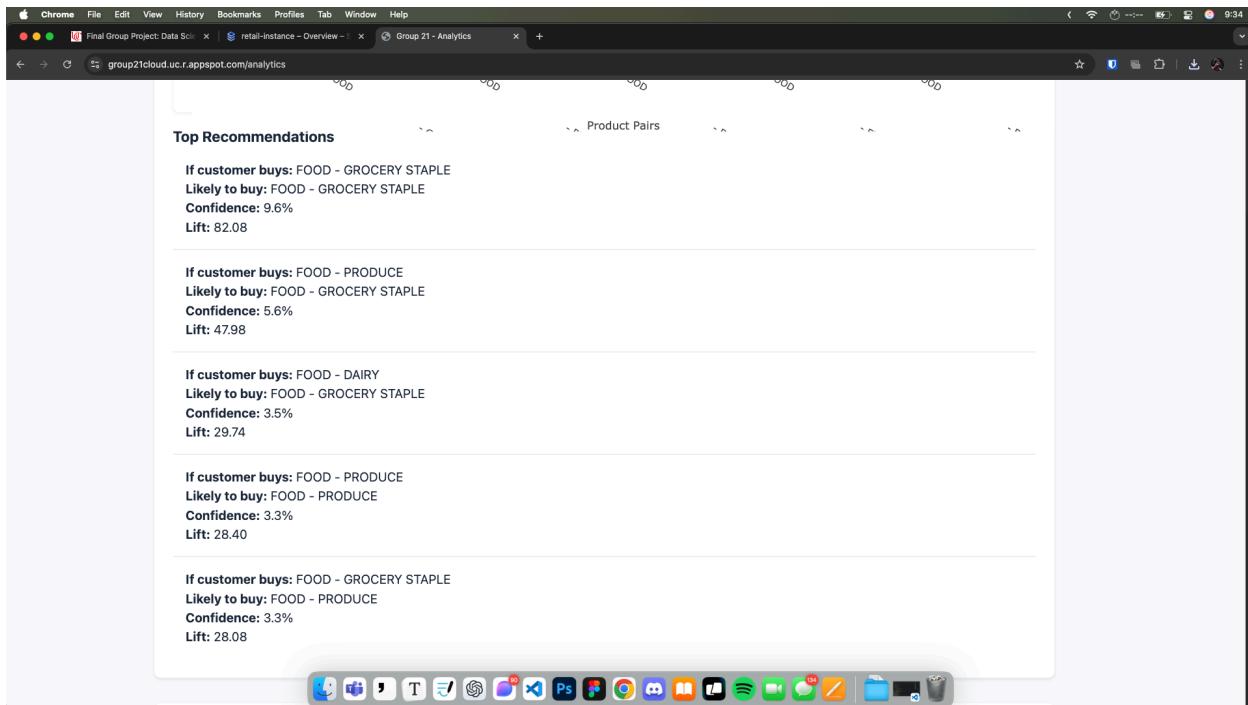
#### Marketing Initiatives:

- Develop bundle offers combining GROCERY STAPLES
- Create promotions linking PRODUCE with GROCERY STAPLES
- Implement targeted recommendations in digital platforms

## Basket Analysis



## Top Recommendations



## 8. Churn Prediction

### Model Implementation

**Model Used:** Random Forest Classifier

**Purpose:** Identify customers at risk of disengaging

#### Performance Metrics:

Model Accuracy: 100%

Churn Rate: 50%

### Customer Risk Distribution

Risk Segmentation:

- High Risk: 198 customers (49.5%)
- Medium Risk: 3 customers (0.75%)
- Low Risk: 49.8% of customer base

Feature Importance Analysis:

- Most Important: Total baskets (0.8 importance score)
- Second Most Important: Unique departments (0.2 importance score)
- Other Factors: Average spend, household size, and children have minimal impact

### Retention Strategy Recommendations

#### High-Risk Customer Interventions:

- Immediate engagement campaigns for 198 identified high-risk customers
- Personalized incentives based on historical purchase patterns
- Direct communication to understand dissatisfaction factors

#### Behavioral Monitoring:

- Focus on basket frequency as the primary indicator
- Track department diversity in customer purchases
- Monitor average spend patterns for early warning signs

#### Retention Program Design:

- Implement loyalty rewards for consistent shopping frequency
- Encourage cross-department shopping through targeted promotions
- Develop special programs for customers showing decreased activity

## Implementation Impact

### Business Benefits:

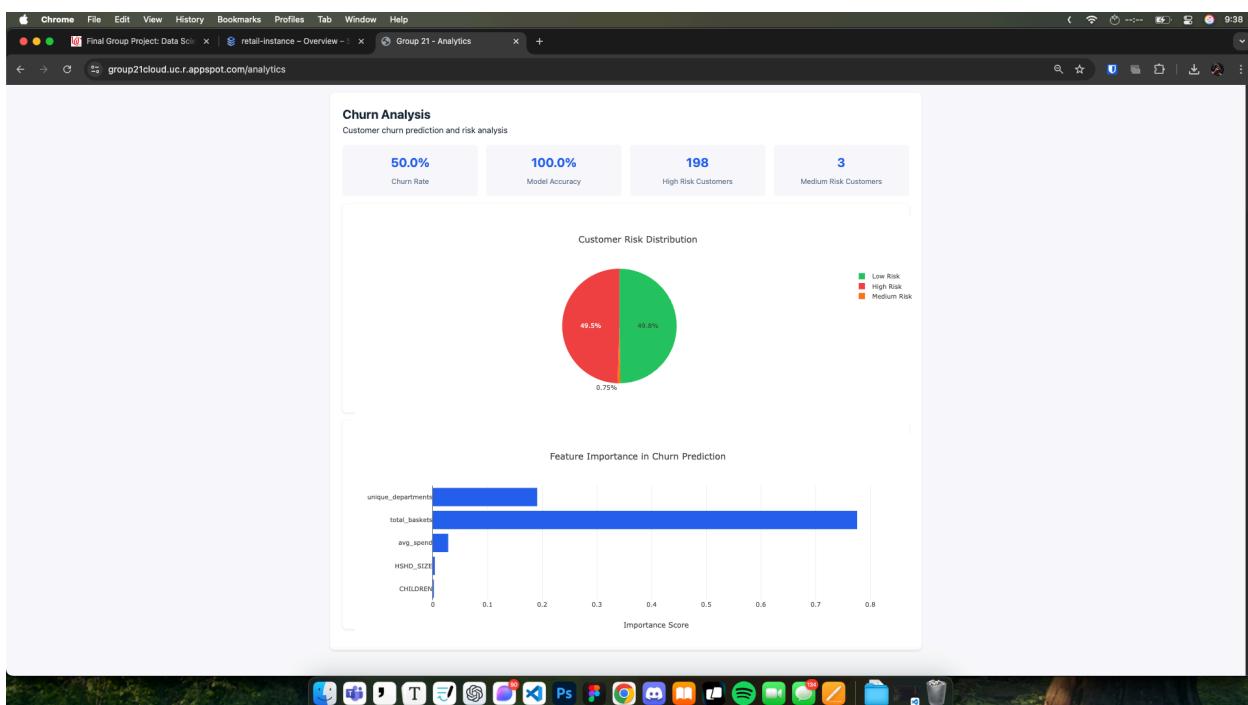
- Enhanced cross-selling through data-driven product placement
- Proactive customer retention through risk identification
- Optimized marketing resource allocation

### Operational Improvements:

- Better inventory management based on product associations
- Targeted marketing campaigns using risk segmentation
- Improved customer engagement through personalized approaches

### Future Enhancements:

- Regular model retraining with new transaction data
- Integration with real-time purchase data for dynamic recommendations
- Development of automated early warning system for churn risk



## Outline of Steps to Implement

This retail analytics project was developed using Google Cloud Platform (GCP) services and Python web technologies to create a data-driven application. The project involves setting up a MySQL database on Cloud SQL, implementing a Flask web application with data visualization capabilities, and incorporating machine learning models for basket analysis and churn prediction, all deployed using Google App Engine.

Key steps included database setup and data loading, web application development with user authentication and search functionality, implementation of interactive dashboards for data visualization, and integration of ML models for retail analytics, culminating in a cloud-based solution for retail business intelligence.

## Step 1. Create and Configure Cloud SQL (MySQL) on GCP

Created a MySQL instance and a database named 'retaildb' with its IP set to public.

The screenshot shows the 'Create a MySQL Instance' page in the Google Cloud Platform. The instance is being configured with the following details:

- Database version:** MySQL 8.0
- Instance ID:** retail-instance
- Password:** admin (GENERATE button)
- Cloud SQL edition:** Enterprise
- Region:** us-central1 (Iowa)
- vCPUs:** 8 vCPU
- RAM:** 32 GB
- Data cache:** Disabled
- Storage:** 250 GB
- Connections:** Public IP
- Backup:** Automated
- Availability:** Multiple zones (highly available)
- Point-in-time recovery:** Enabled
- Network throughput (MB/s):** 2,000 of 2,000
- IOPS:** Read: 7,500 of 15,000  
Write: 7,500 of 15,000
- Disk throughput (MB/s):** Read: 120.0 of 800.0  
Write: 120.0 of 800.0

**Pricing estimate (without discounts):**

Item	Hourly cost (estimate)
8 vCPU (US\$0.041 per vCPU/hour)	US\$0.33
32 GiB RAM (US\$0.007 per GiB/hour)	US\$0.22
250 GiB SSD (US\$0.17 per GiB/month)	US\$0.06
► Standby VM (high availability)	US\$0.61
Total without usage discounts	US\$1.23

**Usage and traffic costs:**

These costs vary based on your feature usage, traffic or data location. [Learn more](#)

**Customise your instance:**

You can also customise instance configurations later

**Show configuration options**

The interface includes a toolbar at the bottom with various application icons.

## Step 2. Create Tables in MySQL

Connected to MySQL using Cloud Shell.

To Activate Cloud Shell I ran this command  
gcloud sql connect retail-instance --user=root

The screenshot shows the Google Cloud SQL Instances page. A single instance, 'retail-instance', is listed. It is an Enterprise edition, MySQL 8.0, with a public IP address of 34.69.93.190. The instance connection name is 'group21cloud:us...'. High availability is enabled, and it is located in us-central1-b. Storage used is 1 GB of 250 GB. There are tabs for 'SQL', 'Instances', 'CREATE INSTANCE', and 'MIGRATE DATABASE'. A search bar at the top right says 'Search (/) for resources, docs, products and more'.

The screenshot shows a Cloud Shell terminal window titled '(group21cloud)'. The MySQL prompt 'mysql>' is visible. The terminal displays the MySQL copyright notice and version information. It also shows the user connecting to the 'retail-instance' database as 'root'.

```
Welcome to Cloud Shell! Type "help" to get started.  
You are currently connected to the instance group21cloud.  
Use "gcloud config set project [PROJECT_ID]" to change to a different project.  
rohanamjeetvee@group21cloud: ~ (group21cloud)$ gcloud sql connect retail-instance --user=root  
Allowlisting your IP for incoming connection for 5 minutes...done.  
Connecting to database with SQL user [root].Enter password:  
Welcome to the MySQL monitor. Commands end with ; or g.  
Your MySQL connection id is 119  
Server version: 8.0.31-google (Google)  
Copyright (c) 2000, 2024, Oracle and/or its affiliates.  
Oracle is a registered trademark of Oracle Corporation and/or its  
affiliates. Other names may be trademarks of their respective  
owners.  
Type 'help,' or 'h' for help. Type 'v' to clear the current input statement.  
mysql> |
```

The screenshot shows the Google Cloud SQL Instances page again, displaying the same 'retail-instance' details as before.

The screenshot shows a Cloud Shell terminal window titled '(group21cloud)'. The MySQL prompt 'mysql>' is visible. The terminal shows the creation of two tables: 'Products' and 'Transactions'. Both queries are successful, with 0 rows affected.

```
--> INCOME_RANGE VARCHAR(50),  
--> HOMEOWNER_DESC VARCHAR(50),  
--> HSBD_COMPOSITION VARCHAR(50),  
--> HSBD_SIZE INT,  
--> CHILDREN INT  
--> );  
Query OK, 0 rows affected (0.09 sec)  
mysql> CREATE TABLE Products {  
--> PRODUCT_NUM INT PRIMARY KEY,  
--> DEPARTMENT VARCHAR(50),  
--> COMMODITY VARCHAR(50),  
--> BRAND_TYPE VARCHAR(50),  
--> NATURAL_ORGANIC_FLAGS VARCHAR(50)  
--> };  
Query OK, 0 rows affected (0.08 sec)  
mysql> CREATE TABLE Transactions {  
--> HSBD_NUM INT,  
--> BASKET_NUM INT,  
--> DATE DATE,  
--> PURCHASE_NUM INT,  
--> SPEND_FLGAT,  
--> UNITS INT,  
--> STORE_REGION VARCHAR(50),  
--> WEEK_NUM INT,  
--> YEAR INT  
--> };  
Query OK, 0 rows affected (0.08 sec)  
mysql> |
```

SQL Code to create tables

```
USE retaildb;
```

```
DROP TABLE IF EXISTS Transactions;  
DROP TABLE IF EXISTS Products;  
DROP TABLE IF EXISTS Households;
```

```
CREATE TABLE Households (  
    HSHD_NUM INT PRIMARY KEY,  
    LOYALTY_FLAG VARCHAR(50),  
    AGE_RANGE VARCHAR(50),  
    MARITAL_STATUS VARCHAR(50),  
    INCOME_RANGE VARCHAR(50),  
    HOMEOWNER_DESC VARCHAR(50),  
    HSHD_COMPOSITION VARCHAR(50),  
    HSHD_SIZE INT,  
    CHILDREN INT  
);
```

```
CREATE TABLE Products (  
    PRODUCT_NUM INT PRIMARY KEY,  
    DEPARTMENT VARCHAR(50),  
    COMMODITY VARCHAR(50),  
    BRAND_TYPE VARCHAR(50),  
    NATURAL_ORGANIC_FLAG VARCHAR(50)  
);
```

```
CREATE TABLE Transactions (  
    HSHD_NUM INT,  
    BASKET_NUM INT,  
    DATE DATE,  
    PRODUCT_NUM INT,  
    SPEND FLOAT,  
    UNITS INT,  
    STORE_REGION VARCHAR(50),  
    WEEK_NUM INT,  
    YEAR INT  
);
```

## Step 3. Load Data into Cloud SQL

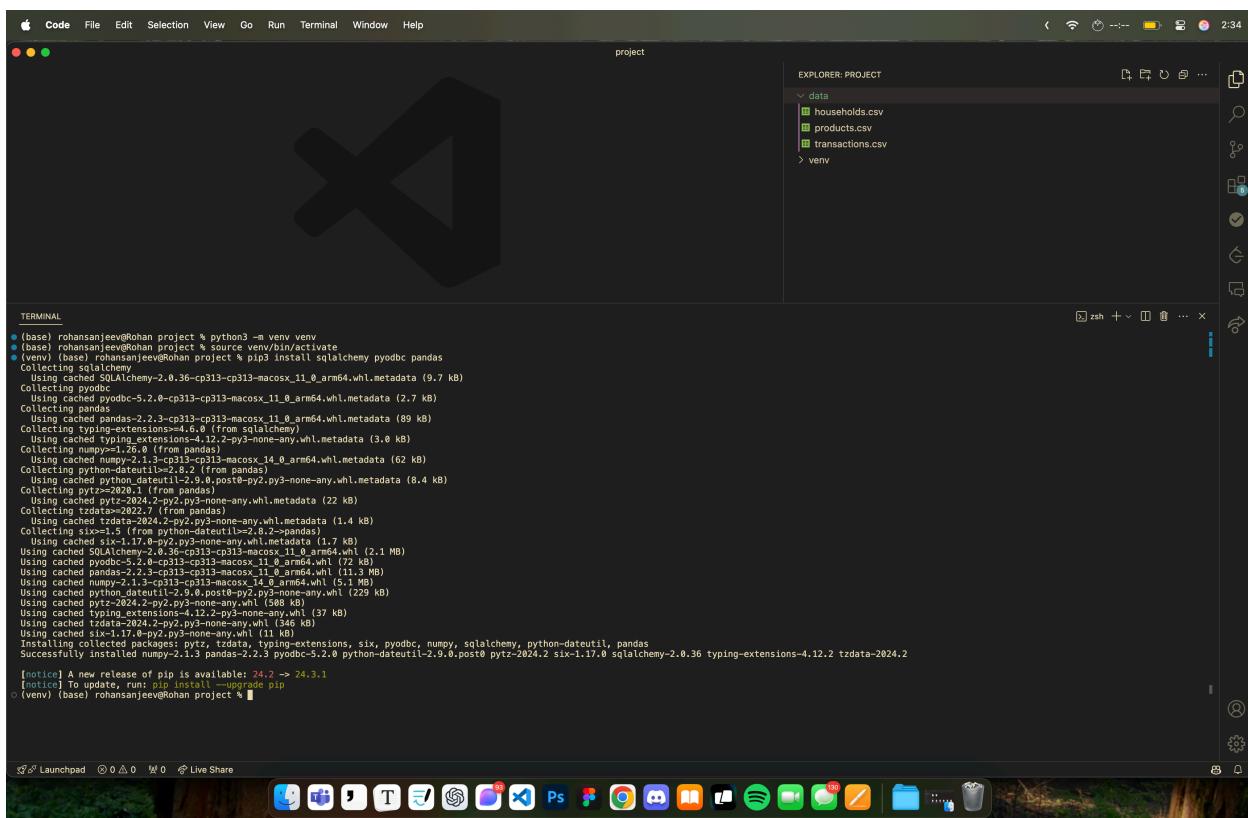
Downloaded the datasets from canvas, create a data folder and place `households.csv`, `products.csv`, `transactions.csv` and created a python virtual environment and installed the dependencies.

To create a virtual environment

```
python -m venv venv  
source venv/bin/activate
```

To install dependencies

```
pip install sqlalchemy pyodbc pandas
```



Create a load\_data.py script

```
import pandas as pd
from sqlalchemy import create_engine, text

DB_USER = "root"
DB_PASS = "admin"
DB_NAME = "retaildb"
DB_HOST = "34.69.93.190"
DB_PORT = 3306

engine = create_engine(f"mysql+pymysql://{DB_USER}:{DB_PASS}@{DB_HOST}:{DB_PORT}/
{DB_NAME}")

households = pd.read_csv("data/households.csv")
products = pd.read_csv("data/products.csv")
transactions = pd.read_csv("data/transactions.csv")

households.columns = [col.strip() for col in households.columns]
products.columns = [col.strip() for col in products.columns]
transactions.columns = [col.strip() for col in transactions.columns]

households.rename(
    columns={
        'L': 'LOYALTY_FLAG',
        'MARITAL': 'MARITAL_STATUS',
        'HOMEOWNER': 'HOMEOWNER_DESC',
        'HH_SIZE': 'HSHD_SIZE',
    },
    inplace=True
)

products.rename(
    columns={
        'BRAND_TY': 'BRAND_TYPE'
    },
    inplace=True
)

transactions.rename(
    columns={
        'PURCHASE_': 'DATE',
        'STORE_R': 'STORE_REGION',
    },
    inplace=True
)

households = households.drop_duplicates(subset=['HSHD_NUM'], keep='first')
products = products.drop_duplicates(subset=['PRODUCT_NUM'], keep='first')
```

```

if 'HSHD_SIZE' not in households.columns:
    print("HSHD_SIZE column not found in households. Adding a placeholder column.")
    households['HSHD_SIZE'] = 0

if 'CHILDREN' not in households.columns:
    print("CHILDREN column not found in households. Adding a placeholder column.")
    households['CHILDREN'] = 0

households['HSHD_SIZE'] = pd.to_numeric(households['HSHD_SIZE'], errors='coerce').fillna(0).astype(int)
households['CHILDREN'] = pd.to_numeric(households['CHILDREN'], errors='coerce').fillna(0).astype(int)

transactions['DATE'] = pd.to_datetime(transactions['DATE'], format='%Y-%m-%d', errors='coerce')

with engine.connect() as conn:
    conn.execute(text("SET FOREIGN_KEY_CHECKS=0;"))
    conn.execute(text("DELETE FROM Transactions;"))
    conn.execute(text("DELETE FROM Products;"))
    conn.execute(text("DELETE FROM Households;"))
    conn.execute(text("SET FOREIGN_KEY_CHECKS=1;"))
    conn.commit()

try:
    # Load in correct order (parents before children)
    households.to_sql('Households', con=engine, if_exists='append', index=False, chunksize=1000)
    print("Households data loaded successfully")

    products.to_sql('Products', con=engine, if_exists='append', index=False, chunksize=1000)
    print("Products data loaded successfully")

    transactions.to_sql('Transactions', con=engine, if_exists='append', index=False, chunksize=1000)
    print("Transactions data loaded successfully")

except Exception as e:
    print(f"Error loading data: {str(e)}")
    raise

print("All data loaded successfully into MySQL on GCP.")

```

## Step 4. Application Structure

```
project/
|   └── data/
|       ├── households.csv
|       ├── products.csv
|       └── transactions.csv
|
|   └── static/
|       └── style.css
|
|   └── templates/
|       ├── login.html
|       ├── search.html
|       ├── dashboard.html
|       ├── upload.html
|       └── analytics.html
|
|   └── venv/
|
|       ├── main.py          # Main application file
|       ├── db_utils.py      # Database utilities
|       ├── ml_utils.py      # ML functionalities
|       ├── data_cleaning.py # Data cleaning utilities
|       ├── requirements.txt # Project dependencies
|       ├── app.yaml         # App Engine configuration
|       └── load_data.py     # Data loading script
```

## Step 5. Set Up Cloud Project Environment

Create a new Google Cloud Platform project  
Enable required APIs (Cloud SQL, App Engine)  
Set up Cloud SQL Auth Proxy (if needed)

## **Step 6. Create Basic Web Application Structure**

Create necessary directories (templates, static)

Create base files (main.py, requirements.txt)

Set up app.yaml for App Engine configuration

## **Step 7. Create Database Connection Utilities**

Create db\_utils.py for database connection handling

Set up environment variables for database credentials

## **Step 8. Implement Core Features**

Login/Authentication system

Search functionality

Dashboard with visualizations

Data upload capability

ML analytics features

## **Step 9. Create Web Templates**

Create HTML templates for each page

Implement CSS styling

Add JavaScript for interactivity

## **Step 10. Set Up ML Pipeline**

Create ml\_utils.py for ML functionality

Implement basket analysis

Implement churn prediction

## **Step 11. Deploy Application**

Test locally first

Update requirements.txt with all dependencies

Configure app.yaml

Deploy to App Engine using: gcloud app deploy

## **Step 12. Post-Deployment Tasks**

- Verify database connections
- Test all functionalities
- Monitor application performance
- Set up logging

## **Step 13. Additional Optimizations**

- Add indexes to database tables
- Optimize SQL queries
- Implement caching if needed
- Add error handling