BLG-317E Database System Project Group 11

Muhammet Serdar NAZLI
Artificial Intelligence and Data Engineering Department
150210723
nazlim21@itu.edu.tr
Istanbul Technical University

Ömer Faruk AYDIN
Artificial Intelligence and Data Engineering Department
150210726
aydinome21@itu.edu.tr
Istanbul Technical University

Mehmet Ali BALIKÇI
Computer and Informatics Engineering Department
150200059
balikci20@itu.edu.tr
Istanbul Technical University

Mehmet Umut GÖKDAĞ
Computer and Informatics Engineering Department
150200085
gokdag20@itu.edu.tr
Istanbul Technical University

Hasan Taha BAĞCI
Artificial Intelligence and Data Engineering Department
150210338
bagcih21@itu.edu.tr
Istanbul Technical University

I. Introduction

The details of the website project, in which the backend, frontend and database parts were coded as a project assignment for the Database Systems course, are detailed under the headings on the following pages.

The aim of the project is to enable easy and fast tracking of owned goods that can be used by companies and tradesmen in particular and by everyone in general. Instead of a complex interface, it has been kept as simple as possible so that operations can be carried out easily and quickly. The website is currently designed to perform basic functions, but is open to improvements. It can be customized according to the area in which it is used, the institution or individuals using it.

It can be integrated with devices such as barcode readers and QR code readers, and by adding advanced features such as detecting the photo of the product or goods, the website can be made more functional and useful depending on the area in which it is used.

We did the project as a group of 5 students. Those who carried out the project and their duties are listed below.

- Muhammet Serdar Nazlı - Database operations, Backend operations
- Ömer Faruk Aydın - Frontend operations, UI views
- Hasan Taha Bağcı - Backend operations optimization
- Mehmet Ali Balıkçı - Frontend operations and UI views
- Mehmet Umut Gökdağ - Backend operations optimization

II. Personal Contributions

Muhammet Serdar NAZLI

- I was solely responsible for managing the Inventory Items table, which included overseeing its design, ensuring data integrity, and regularly updating the table to reflect current inventory status.
- Played a pivotal role in developing initial prototypes, utilizing Flask and MySQL. This involved designing the fundamental architecture of the website, establishing database connections, and implementing initial functionalities.
- Contributed significantly to backend operations, particularly focusing on CRUD (Create, Read, Update, Delete) operations. This role also entailed developing and maintaining robust connections between the frontend and backend, ensuring seamless data flow and user interaction.
- Assisted in the process of loading the dataset onto the MySQL server. This task required meticulous attention to data format and structure, ensuring the integrity of data during transfer.

Ömer Faruk AYDIN

- Management and maintenance of the ”Order Items” table, which involved ensuring data integrity and accuracy within this crucial segment of the database.
- Design and configuration of the Web UI, focusing on optimizing user experience and ensuring the interface is interactive and user-friendly.

- Handling backend operations using JavaScript, which encompassed processing data and implementing server-side logic to support the application's functionality.
- Coordination and execution of table operations for "Order Items," including efficient management of data updates and queries, ensuring smooth data handling and retrieval processes.

Mehmet Ali BALIKÇI

- My primary responsibility was the management and upkeep of the 'products' table in our database. This role involved not only maintaining the data within this table but also ensuring it was consistently up-to-date and accurately reflected our product inventory.
- I played a key role in resolving data integrity issues, which involved identifying and rectifying discrepancies in our database. This task was crucial to maintain the reliability and accuracy of our data, which is fundamental to the overall functionality of our database systems.
- I was responsible for writing and implementing a function to retrieve data from various tables in the database. This function was essential for facilitating smooth data access and manipulation, thereby enhancing the efficiency of our database operations.
- I oversaw the frontend development of the products page. This included designing the user interface, ensuring it was intuitive and user-friendly, and implementing the front-end logic to interact effectively with our backend systems, providing a seamless user experience.

Mehmet Umut GÖKDAĞ

- I was primarily in charge of overseeing and maintaining the 'events' table within our database.
- I was instrumental in solving data integrity problems by detecting and correcting mismatches in our database. This effort was vital to ensure the dependability and precision of our data, which is essential for the proper operation of our database systems.
- My role involved creating and executing a function to fetch information from different tables within the database. This function played a vital role in ensuring seamless data retrieval and manipulation, ultimately improving the overall efficiency of our database operations.

Hasan Taha BAĞCI

- I began by adding the orders file to the tables folder. This step was crucial as it laid the foundation for the database structure we needed for managing ecommerce orders.
- Utilizing MySQL Workbench, I created the SCHEMA.sql file. This involved carefully designing the database schema to ensure efficient data storage and retrieval, which is essential for handling ecommerce transactions.
- I developed classes and functions for the orders' table, which were incorporated into the utils/table_operations.py file. This code is the backbone of our order management system, handling everything from order placement to tracking and processing.
- In order to facilitate the smooth operation of our database, I created the creat_db.py file. This script automates the importation of our database schema into a local MySQL setup, ensuring that our team can easily set up their development environments.
- To improve user experience, I updated the HTML file with additional CSS and JavaScript. These add-ons were not just about aesthetics; they were implemented to make the user interface more intuitive and responsive.
- Updated the README.md file to reflect these changes and additions. This documentation is vital for both our current team and future developers, providing clear guidance on how the system operates and how to work with it.

III. Dataset Outline

The dataset for our project is structured into seven primary tables, each representing a different aspect of our e-commerce platform:

A. Users

- Contains personal information about users such as name, email, and address.
- Includes geographic information like state, city, and postal code.
- Records user activity data like browser and traffic source.

B. Events

- Tracks user actions within the system with details such as session IDs and event types.
- Includes the timestamp of each event for chronological analysis.

C. Products

- Catalogs products with attributes like cost, category, name, and brand.
- Includes retail price and the associated distribution center.

D. Orders

- Details each order made by users, including status and dates for creation, shipment, and delivery.
- Stores the number of items per order.

E. Order Items

- Lists individual items within orders, linking to both the order and the product.
- Captures the sale price and status at the time of order.

F. Inventory Items

- Keeps track of inventory levels, with timestamps for when items are added or sold.
- Contains detailed product information replicated from the Products table for quick reference.

G. Distribution Centers

- Provides information on various distribution centers including their geographic coordinates.
- Is linked to the Products and Inventory Items tables.

IV. Database Connection and Initialization

The Flask application's interaction with the MySQL database involves two key aspects: establishing a connection and initializing the database. This section outlines the configuration, connection establishment, and initialization process.

A. Configuration

The database connection parameters are defined in a separate configuration file, 'settings.py'. This file contains the necessary credentials such as the database user, password, host, and database name, which are crucial for a secure and reliable connection.

```
db_user = 'your_database_username'
db_password = 'your_database_password'
db_host = 'your_database_host'
db_name = 'your_database_name'
#Suggested db_name=thelook_ecommerce
```

B. Establishing Connection

The 'app.py' file initiates a connection to the MySQL database using these parameters, employing the 'mysql.connector.connect()' function.

```
connection = mysql.connector.connect(
                        host=db_host,
                        database=db_name,
                        user=db_user,
                        password=db_password)
```

C. Database Initialization

Before running the application, the database is set up using the 'create_db.py' script. This script creates the database and initializes its schema.

```
mydb = mysql.connector.connect(
    host=db_host,
    user=db_user,
    password=db_password,
    auth_plugin='mysql_native_password'
)
mycursor.execute(f"CREATE DATABASE {db_name}")
```

The script then executes SQL commands from 'schema.sql' to create tables and other structures necessary for the application.

```
cnx = mysql.connector.connect(
    user=db_user,
    password=db_password,
    host=db_host,
    database=db_name,
    auth_plugin='mysql_native_password')
...
executeScriptsFromFile('./database/schema.sql')
```

D. Operations on Database

The connection that is created on 'app.py' file (see subsection B.) is passed to the proper classes and then The Flask application performs various operations on the database:

- **Inserting Data:** New records are inserted into the database using the 'insert_data' method of the respective table class.
- **Updating Data:** Existing records are updated using the 'update_data' method.
- **Deleting Data:** Records are deleted using the 'delete_data' method.
- **Searching Data:** The application supports searching for records in the database using the 'search' method.

Each class encapsulates the logic for interacting with its corresponding table, thus maintaining a clean and organized code structure.

E. Session Management

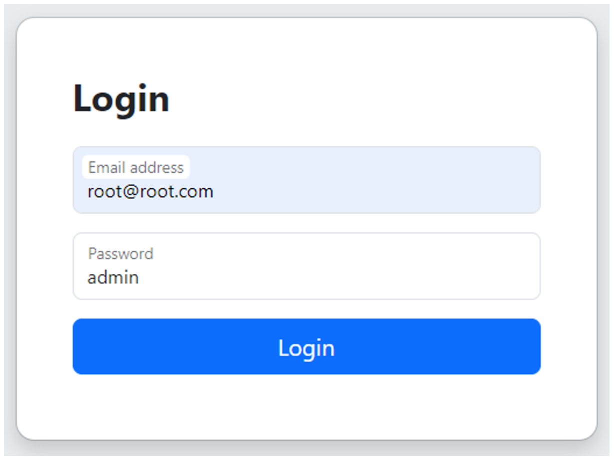
Flask's session management is used to control access to the database. Users must log in to interact with the database, ensuring unauthorized users cannot access or modify data.

```
@app.route('/login', methods=['GET', 'POST'])
def login():
    ...
```

In summary, the database connection and initialization in our Flask application are designed to be secure, efficient, and modular, enabling effective data management.

V. Website

Login Page


Login

Email address
root@root.com

Password
admin

Fig. 1. Login Page

Our website works as a kind of admin panel and certain admin usernames and passwords must be used to log in to the system. As shown in the figure1, the username is ”root@root.com” and the password is ”admin”.

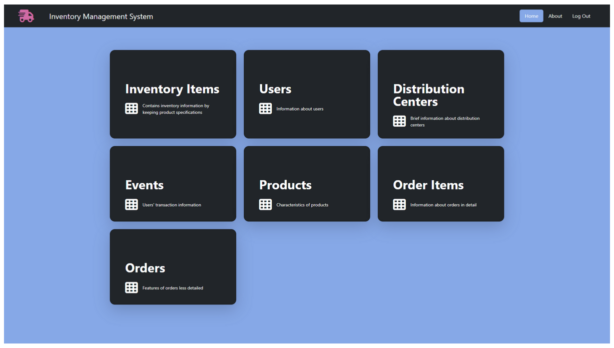

Inventory Management System
Home
About
Log Out
Inventory Items
Contains inventory information by keeping product specifications
Users
Information about users
Distribution Centers
Brief information about distribution centers
Events
Users' transaction information
Products
Characteristics of products
Order Items
Information about orders in detail
Orders
Features of orders less detailed

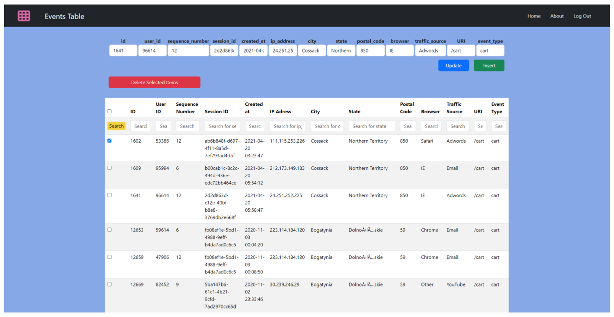
Fig. 2. Home Page

Home Page

As can be seen in the figure5, there are 7 tables on our Home screen page. 5 of them are main tables (Inventory Items, Orders, Events, Products and Order Items), 2 of them are common tables (Users, Distribution Centers).

Table View

Basically all tables perform similar operations. Table pages consist of similar components. There is a section on the table



id	user_id	sequence_number	session_id	created_at	ip_address	city	state	postal_code	browser	traffic_source	URI	event_type
1641	96614	12	2d2d863c	2021-04-:	24.251.25	Cossack	Northern	850	IE	Adwords	/cart	cart

	ID	User ID	Sequence Number	Session ID	Created at	IP Adress	City	State	Postal Code	Browser	Traffic Source	URI	Event Type
Search	Searcl	Sea	Search	Search for se	Searc	Search for ip_	Search for c	Search for state	Sea	Searcl	Search	Se	Sea
✓	1602	53386	12	ab6b848f-d697-4f11-8a5d-7ef793ad4dbf	2021-04-20 03:23:47	111.115.253.226	Cossack	Northern Territory	850	Safari	Adwords	/cart	cart
☐	1609	95994	6	b00cab1c-8c2c-494d-936e-edc72bb464ce	2021-04-20 05:54:12	212.173.149.183	Cossack	Northern Territory	850	IE	Email	/cart	cart
☐	1641	96614	12	2d2d863d-c12e-40bf-b8e8-3769db2e668f	2021-04-20 05:58:47	24.251.252.225	Cossack	Northern Territory	850	IE	Adwords	/cart	cart
☐	12653	59614	6	fb08ef1e-5bd1-4988-9eff-b4da7ad0c6c5	2020-11-03 00:04:20	223.114.184.120	Bogatynia	DolnoÅ›lÄ…skie	59	Chrome	Email	/cart	cart
☐	12659	47906	12	fb08ef1e-5bd1-4988-9eff-b4da7ad0c6c5	2020-11-03 00:08:50	223.114.184.120	Bogatynia	DolnoÅ›lÄ…skie	59	Chrome	Email	/cart	cart
☐	12669	82452	9	5ba147b6-61c1-4b21-9cfd-7ad2970cc65d	2020-11-02 23:33:46	30.239.246.29	Bogatynia	DolnoÅ›lÄ…skie	59	Other	YouTube	/cart	cart

Fig. 3. Table View

for Insert and Update operations6. For the Update operation, click on the row to be updated and the relevant information goes to the relevant boxes at the top. Then the columns to be updated are changed and the update button is pressed. Insert operation is provided with a similar process. the relevant boxes are filled and the insert button is pressed. A third operation is the Search operation. Search operation can be done by giving more than one parameter. After filling the relevant columns where the search will be performed, the search button is pressed and the data returned from the database is returned to the screen. Delete operation can be used as the last operation. Select the box at the beginning of the rows to be deleted or if you want to delete all rows, select the top box (Select all) and click on Delete Selected Items button. If you want to delete specific rows, you can first find these rows with the search operation and then delete all operation can be applied.

Queries - CRUD Operations

The website has been designed in such a way that Create, Read (Select), Update and Delete operations can be performed on all tables on it. The execution of these operations on the tables is done through classes and methods, thus ensuring compliance with the encapsulation principle.

While performing each of the following operations, an appropriate query is sent to the database via the cursor.

While performing the insertion process, a primary key is generated through an auxiliary function, and then the line containing that primary key is filled with the data entered by the user. The data entered by the user is kept in a list and added to the table sequentially according to column names.

The search process allows the user to search by any column in the table. The user can access the target row with any information he wants. Performing the search process in this way is an important feature to speed up the process.

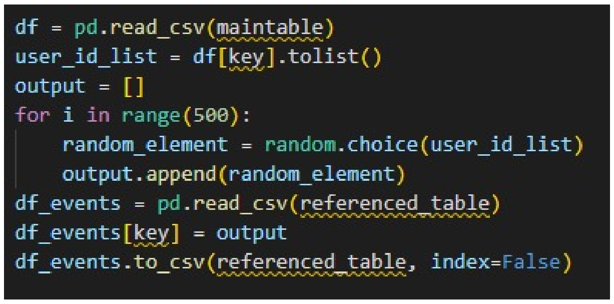
The update process is done by asking the user for an ID number and entering this ID number. After the user enters his ID number, the line he wants to change appears, and at this stage, the user does not have to re-enter all the information in the line. He can change any value in any column he wants. The reason why the update process is done only based on the ID number is to gain speed. After entering the ID number, a single line will appear before the user and the user will change this line. If updates could be made with other columns, the user would have to find the row he is looking for among many options.

The delete operation is performed similarly to the search operation. The user can delete single or multiple lines by checking the boxes next to the lines appearing on the page after or without searching.

This is the background of the transactions and operations performed on the tables.

VI. DIFFICULTIES

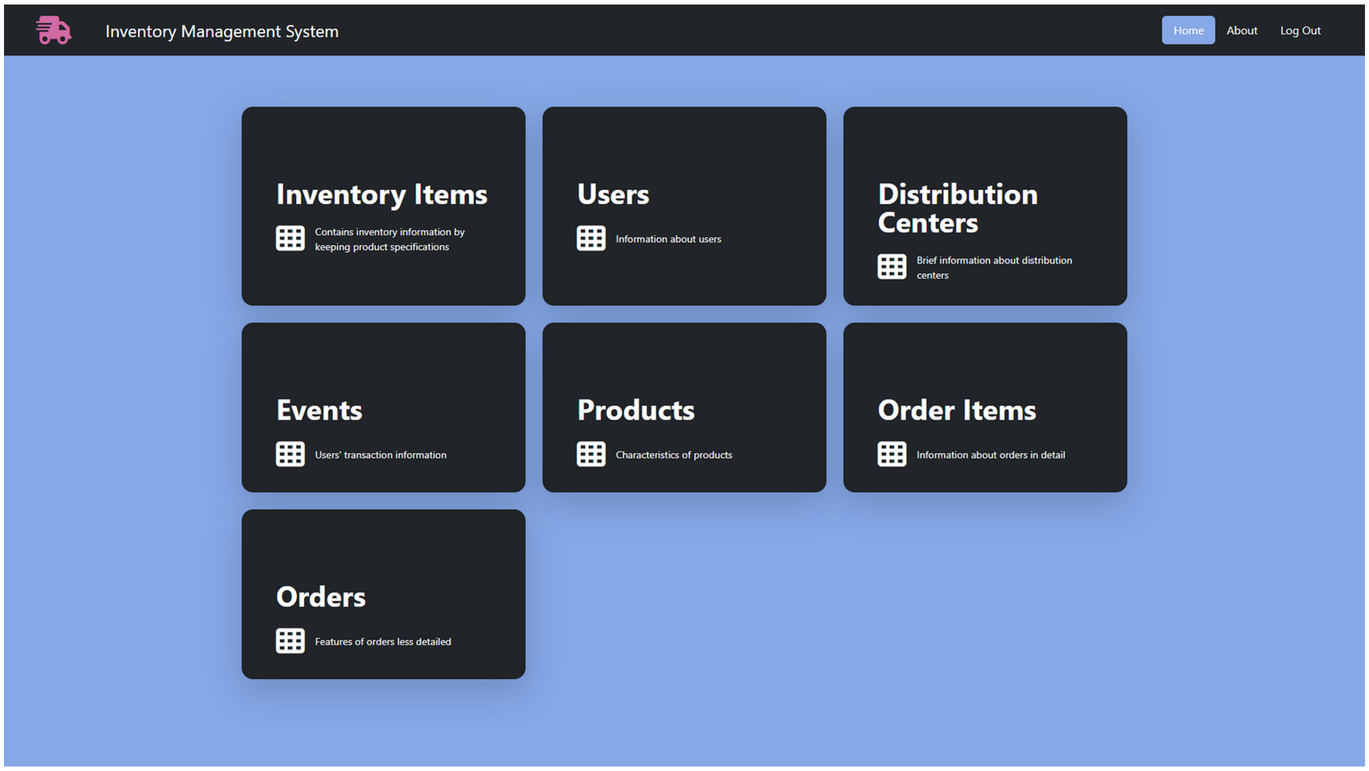
There were integrity problems caused by the mismatch between the foreign keys in the tables and the primary keys in the main table to which they are referenced, and we solved these problems with the provided python code.



```
df = pd.read_csv(maintable)
user_id_list = df[key].tolist()
output = []
for i in range(500):
    random_element = random.choice(user_id_list)
    output.append(random_element)
df_events = pd.read_csv(referenced_table)
df_events[key] = output
df_events.to_csv(referenced_table, index=False)
```

Fig. 4. Code Block

This Python code reads a CSV file named 'orders.csv' from a folder 'tables/thelook-ecommerce' and extracts the 'order-id' column into a list. Then, it randomly selects 500 elements from this list, generating a new list. Next, it reads another CSV file named 'order-items.csv' from the same folder. The code replaces the 'order-id' column in this second DataFrame with the randomly generated list of order IDs. Finally, it saves the modified DataFrame back to ’order-items.csv’, overwriting the original file, and does not include the DataFrame index in the saved CSV file.



Inventory Items
Contains inventory information by keeping product specifications

Users
Information about users

Distribution Centers
Brief information about distribution centers

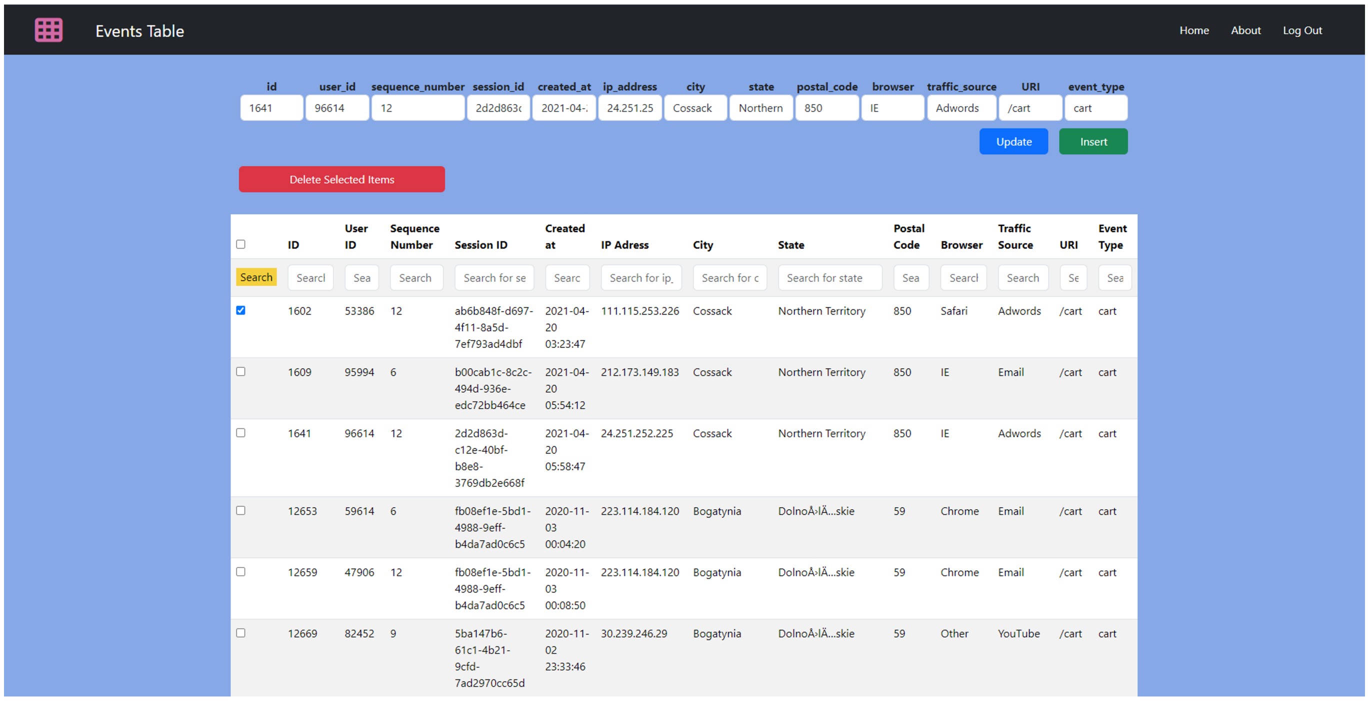
Events
Users' transaction information

Products
Characteristics of products

Order Items
Information about orders in detail

Orders
Features of orders less detailed

Fig. 5. Home Page



id	user_id	sequence_number	session_id	created_at	ip_address	city	state	postal_code	browser	traffic_source	URI	event_type
1641	96614	12	2d2d863c	2021-04-:	24.251.25	Cossack	Northern	850	IE	Adwords	/cart	cart

	ID	User ID	Sequence Number	Session ID	Created at	IP Adress	City	State	Postal Code	Browser	Traffic Source	URI	Event Type
	Searcl	Sea	Search	Search for se	Searc	Search for ip_	Search for c	Search for state	Sea	Searcl	Search	Se	Sea
✓	1602	53386	12	ab6b848f-d697-4f11-8a5d-7ef793ad4dbf	2021-04-20 03:23:47	111.115.253.226	Cossack	Northern Territory	850	Safari	Adwords	/cart	cart
☐	1609	95994	6	b00cab1c-8c2c-494d-936e-edc72bb464ce	2021-04-20 05:54:12	212.173.149.183	Cossack	Northern Territory	850	IE	Email	/cart	cart
☐	1641	96614	12	2d2d863d-c12e-40bf-b8e8-3769db2e668f	2021-04-20 05:58:47	24.251.252.225	Cossack	Northern Territory	850	IE	Adwords	/cart	cart
☐	12653	59614	6	fb08ef1e-5bd1-4988-9eff-b4da7ad0c6c5	2020-11-03 00:04:20	223.114.184.120	Bogatynia	DolnoÅ›lÄ…skie	59	Chrome	Email	/cart	cart
☐	12659	47906	12	fb08ef1e-5bd1-4988-9eff-b4da7ad0c6c5	2020-11-03 00:08:50	223.114.184.120	Bogatynia	DolnoÅ›lÄ…skie	59	Chrome	Email	/cart	cart
☐	12669	82452	9	5ba147b6-61c1-4b21-9cfd-7ad2970cc65d	2020-11-02 23:33:46	30.239.246.29	Bogatynia	DolnoÅ›lÄ…skie	59	Other	YouTube	/cart	cart

Fig. 6. Table View