

CPSC 304 Project Cover Page

Milestone #: 4

Date: July 28, 2022

Group Number: 22

Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Luis Victoria	78827979	x1p2b	luisv@student.ubc.ca
Richard Chen	45564895	n4z2o	richard.chen@ualberta.ca
Yan Zhang	33283136	l8b3b	yyzhang@student.ubc.ca

By typing our names and student numbers in the above table, we certify that the work in the attached assignment was performed solely by those whose names and student IDs are included above. (In the case of Project Milestone 0, the main purpose of this page is for you to let us know your e-mail address, and then let us assign you to a TA for your project supervisor.)

In addition, we indicate that we are fully aware of the rules and consequences of plagiarism, as set forth by the Department of Computer Science and the University of British Columbia

Repository Link

https://github.students.cs.ubc.ca/CPSC304-2022S-T2/project_l8b3b_n4z2o_x1p2b

Short Description + Accomplishment

The project, Bitlink, is a payment processor that allows customers to pay for products using Bitcoin while merchants receive US Dollars. Customers fund their Bitlink accounts beforehand and can then make purchases from accepted vendors. Bitlink acts as a middleman and will convert the customer's Bitcoins into USD using the exchange rate at the time of purchase (shown at the checkout page).

Additionally, customers and merchants have a dashboard where they are able to see and filter their past transactions.

Description of final schema vs schema turned in originally

Wallet(wallet_id: integer, btc_amount: double)
PK: wallet_id

Withdrawal(transaction_id: integer, customer_btc_address: string, btc_to_withdraw: double)
PK: transaction_id

Deposit(transaction_id: integer, bitlink_btc_address: string)
PK: transaction_id

Transaction(transaction_id: integer, **wallet_id**: integer)
PK: transaction_id
FK: wallet_id references InternalEWallet

Customer(customer_id: integer, **wallet_id**: integer, name: string, email: string)
PK: customer_id
FK: wallet_id references InternalEWallet
CKs: customer_id, email

OrderDetails(order_id: integer, **customer_id**: integer, **company_account_number**: integer, **merchant_id**: integer, **wallet_id**: integer, datetime: date, fee_percentage: double)
PK: order_id
FK: customer_id references Customer
company_account_number references CompanyAccount(account_number)

merchant_id references Merchant
wallet_id references Wallet

Merchant(merchant_id: integer, **bank_account_number**: integer, name: string,
usd_owed: double)
 PK: merchant_id
 FK: bank_account_number references
 MerchantBankAccount(account_number)

MerchantBankAccount(account_number: integer, routing_number: integer)
 PK: account_number, routing_number

CompanyAccount(account_number: integer, usd_balance: double, btc_balance: double)
 PK: account_number

Subscription(order_id: integer, conversion_rate: double, charge_usd_price: double,
billing_frequency: string, billing_duration: integer)
 PK: order_id

OnetimePurchase(order_id: integer, conversion_rate: double, total_usd_price: double)
 PK: order_id

LineItemType(order_id: integer, item_name: string, item_type: string)
 PK: order_id, item_name
 FK: order_id references OrderDetails

LineItem(order_id: integer, item_brand: string, **item_name**: string, item_usd_price:
double, item_quantity: integer)
 PK: order_id, item_brand, item_name
 FK: order_id references OrderDetails
 item_name references LineItemType

The final schema turned out to be very similar to one which was turned in for Milestone 2. Besides from the renaming of some tables to have clearer names, and changes to some attributes types like billing_duration in Subscription to be integer instead of string, only one table was dropped, MerchantBankAddress, since that was created out of an inefficiency meant only for the purposes of a normalization exercise to meet Milestone 2 criteria.

List of SQL queries used

Important Notes

Line Number refers to where you can find the function in the `queries.js` file

URLs are displayed after the BASE URL. So if your base URL is `localhost:8080` and you are trying to use `getWallets` with the URL `/wallet`, then you access it by visiting `localhost:8080/wallet` as a GET request

Some queries require the use of parameters. To access `orderProj` for example, you need to use a GET request with the URL `/orderProj?order_id=true&customer_id=false&...`

`${variable}` notation indicates users are able to specify input values

GET Requests

Function Name	<code>getWallets</code>
Query	<code>SELECT * FROM Wallet ORDER BY wallet_id ASC;</code>
URL	<code>/wallet</code>
Line Number	15
Description	Obtain a record of a single wallet

Function Name	<code>getWallet</code>
Query	<code>SELECT * FROM Wallet WHERE wallet_id = \$1</code>
URL	<code>/wallet/:id</code>
Line Number	27
Description	Obtain all records of wallets

Function Name	getItemType
Query	SELECT * FROM LineItemType;
URL	/itemType
Line Number	54
Description	Obtain all records of Item Types

Function Name	getCustomers
Query	SELECT * FROM Customer ORDER BY customer_id ASC;
URL	/customer
Line Number	42
Description	Obtain all records of Customers

Function Name	getCustomerByID
Query	SELECT * FROM Customer WHERE customer_id = \$1
URL	/customer/:id
Line Number	63
Description	Obtain the record of the Customer with specified ID

Function Name	getMerchants
Query	SELECT * FROM Merchant ORDER BY merchant_id ASC;
URL	/merchant
Line Number	78
Description	Obtain all records of Merchants

<i>Function Name</i>	getMerchantByID
<i>Query</i>	SELECT * FROM Customer WHERE merchant_id = \$1
<i>URL</i>	/merchant/:id
<i>Line Number</i>	90
<i>Description</i>	Obtain the record of the Merchant with specified ID

<i>Function Name</i>	getOrders
<i>Query</i>	SELECT * FROM OrderDetails ORDER BY order_id ASC;
<i>URL</i>	/order
<i>Line Number</i>	105
<i>Description</i>	Obtain all records of Orders

<i>Function Name</i>	getOrdersProjection *REQUIRED QUERY (Projection Operation)
<i>Query</i>	SELECT <i>order_id, customer_id, company_account_number, merchant_id, wallet_id, datetime, fee_percentage</i> FROM OrderDetails;
<i>URL</i>	/orderProj
<i>Line Number</i>	131
<i>Description</i>	Return records from OrderDetails with the attributes specified by the user in SELECT. Italicized attributes represent the user's ability to include or exclude those attributes in the return of the query.

Before Order Projection Operation

Filter Columns:

Order Number

Customer ID

Company Account Number

Merchant ID

Wallet ID

Date Time

Fee Percentage

☒

☒

☒

☒

☒

☒

☒

Orders

Order_id	Customer_id	Company_account_nu...	Merchant_id	Wallet_id	Datetime	Fee_percentage
1	4	1000000001	3	4	2022-07-31T07:00:00....	0.02
2	1	1000000001	2	1	2022-08-01T07:00:00....	0.05
3	2	1000000002	1	2	2022-08-01T07:00:00....	0.01
4	4	1000000004	4	4	2022-08-03T07:00:00....	0.02
5	4	1000000001	3	4	2022-08-06T07:00:00....	0.02
6	1	1000000005	5	1	2022-07-31T07:00:00....	0.02
7	2	1000000005	5	2	2022-08-01T07:00:00....	0.02
8	3	1000000005	5	3	2022-08-01T07:00:00....	0.02
9	4	1000000005	5	4	2022-08-03T07:00:00....	0.02

Cancel Order

After Order Projection Operation

Filter Columns:

Order Number

Customer ID

Company Account Number

Merchant ID

Wallet ID

Date Time

Fee Percentage

☒

☒

☐

☒

☐

☒

☐

Orders

Order_id	Customer_id	Merchant_id	Datetime
1	4	3	2022-07-31T07:00:00....
2	1	2	2022-08-01T07:00:00....
3	2	1	2022-08-01T07:00:00....
4	4	4	2022-08-03T07:00:00....
5	4	3	2022-08-06T07:00:00....
6	1	5	2022-07-31T07:00:00....
7	2	5	2022-08-01T07:00:00....
8	3	5	2022-08-01T07:00:00....
9	4	5	2022-08-03T07:00:00....

Cancel Order

Function Name	getLineItems
Query	SELECT * FROM LineItem ORDER BY order_id ASC;
URL	/lineitem
Line Number	175
Description	Obtain all records of Items

Function Name	getOnetimePurchase
Query	SELECT * FROM OnetimePurchase ORDER BY order_id ASC;
URL	/otp
Line Number	187
Description	Obtain all records of One-time purchases

Function Name	getSubscription
Query	SELECT * FROM Subscription ORDER BY order_id ASC;
URL	/subscription
Line Number	199
Description	Obtain all records of Subscription purchases

Function Name	getTransactions
Query	SELECT * FROM Transaction ORDER BY transaction_id ASC;
URL	/transaction
Line Number	308
Description	Obtain all records of Transactions

Function Name	getDeposits
Query	SELECT * FROM Deposit ORDER BY transaction_id ASC;
URL	/deposit
Line Number	320
Description	Obtain all records of Deposits

Function Name	getWithdrawals
Query	SELECT * FROM Withdrawal ORDER BY transaction_id ASC;
URL	/withdrawal
Line Number	332
Description	Obtain all records of Withdrawals

Function Name	getPurchaseSelection *REQUIRED QUERY(Selection Operation)
Query	<p>SELECT order_id, conversion_rate, total_usd_price FROM OnetimePurchase WHERE total_usd_price < \${params.priceLessThan}</p> <p>SELECT order_id, conversion_rate, charge_usd_price, billing_frequency, billing_duration FROM Subscription WHERE charge_usd_price > \${params.priceGreaterThan}</p> <p>The above are examples. See Line 216 of queries.js to see how the query string is dynamically built based on user input.</p>
URL	/purchaseSelection
Line Number	216
Description	Return records from the OnetimePurchase or Subscription table with the attributes specified by the user, according to the greater than or less than filter conditions and values if any.

Before Purchase Selection Operation

Transaction Type

One-time Purchase

Filter Columns: ☒ Order ID ☒ Conversion Rate ☒ Price

Filter Price: ☐

Order_id	Conversion_rate	Total_usd_price
1	0.00004304037	30
2	0.00004504037	500
3	0.00003904037	456
4	0.00005704037	1230
5	0.00003704037	256
11	0.00004504037	500
12	0.00003904037	456
13	0.0000418200066912...	100

After Purchase Selection Operation

After swapping to Subscription table, selecting on three attributes, and filtering on orders greater than \$50.

Transaction Type

Subscription Purchase

Filter Columns: ☒ Order ID ☐ Conversion Rate ☒ Price ☒ Billing Frequency ☐ Billing Duration

Filter Price: ☒ Greater Than ☐ Less Than 50

CLEAR

FILTER

Order_id	Charge_usd_price	Billing_frequency
7	56	Monthly
9	100	Monthly
10	80	Yearly

Function name	getLineItemJoin *REQUIRED QUERY (Join Operation)
Query	SELECT O.order_id, M.name, O.datetime, L.item_name, L.item_usd_price, L.item_quantity FROM OrderDetails O, LineItem L, Merchant M WHERE O.order_id = L.order_id AND O.merchant_id = M.merchant_id AND O.customer_id = \${params.customer_id} AND O.datetime = '\${params.date}'
URL	/lineitemJoin?customer_id=4&date=2022-08-07
Line Number	284
Description	Obtain all line item records belonging to a specific customer and optionally on a specific date.

Before Line Item Join Operation

Joins information from the OrderDetails, LineItem, and Merchant tables.

Items Bought

YYYY-MM-DD

FILTER DATE

Order_id	Name	Datetime	Item_name	Item_usd_price	Item_quantity
1	Zara	2022-07-31T07:00:00....	Dress Shirt	30	1
4	Apple	2022-08-03T07:00:00....	Macbook	1230	1
5	Zara	2022-08-06T07:00:00....	Scarf	256	1
9	Netflix	2022-08-03T07:00:00....	Two Month Subscription	100	1
13	Zara	2022-08-08T07:00:00....	MockItem1	100	2

After Line Item Join Operation

Filtering on the date of the order

Items Bought

YYYY-MM-DD

2022-08-03

FILTER DATE

Order_id	Name	Datetime	Item_name	Item_usd_price	Item_quantity
4	Apple	2022-08-03T07:00:00....	Macbook	1230	1
9	Netflix	2022-08-03T07:00:00....	Two Month Subscription	100	1

Function name	getMerchantsAtLeastTwoOrders *REQUIRED QUERY (Aggregation with Having)
Query	<pre> SELECT DISTINCT Merchant.name FROM OrderDetails, Merchant WHERE OrderDetails.merchant_id = Merchant.merchant_id AND OrderDetails.merchant_id IN (SELECT merchant_id FROM OrderDetails GROUP BY merchant_id HAVING COUNT(*) > 1) </pre>
URL	/merchantsAtLeastTwoOrders
Line Number	370
Description	Returns the name of Merchants that have at least two orders in OrderDetails.

After Display Merchant with at least two Orders Aggregation with Having Operation

Transactions

Merchants

Ministry of Supply

Tesla

Zara

Netflix

Aggregation with Having:
Show merchants with at 2 orders

1

2

3

4

Function name	getMostPopularItemType *REQUIRED QUERY (Nested Aggregation with Group By)
Query	<pre> SELECT COUNT(IT.item_type), IT.item_type FROM LineItemType IT GROUP BY IT.item_type HAVING COUNT(IT.item_type) >= ALL (SELECT COUNT(IT2.item_type) FROM LineItemType IT2 GROUP BY IT2.item_type) </pre>
URL	/mostPopularItemType
Line Number	393
Description	Groups LineItemType by the item_type, and determines the one with the highest count as the most popular one.

After Most Popular Item Type Nested Aggregation with Group By Transactions

Item Type	Count
Clothes	7
<div>Nested Aggregation with Group By: Get most popular item type</div>	

1

2

3

4

<i>Function name</i>	getCustomerBoughtAllMerchant *REQUIRED QUERY (Division Operation)
<i>Query</i>	<pre>SELECT c.name FROM Customer c WHERE NOT EXISTS ((SELECT m.merchant_id FROM Merchant m) EXCEPT (SELECT o.merchant_id FROM OrderDetails o WHERE o.customer_id = c.customer_id))</pre>
<i>URL</i>	/customerBoughtAllMerchant
<i>Line Number</i>	412
<i>Description</i>	Finds customers that have placed an Order with every Merchant.

After Determining Customers that bought from every Merchant Division Operation Transactions

Name

Jordon Johnson

Division:

Find customers that have bought from every merchant

1

2

3

4

PUT Requests

Function name	updateWallet
Query	UPDATE Wallet SET btc_amount = \$1 WHERE wallet_id = \$2
URL	/wallet/:id
Line Number	439
Description	Updates the customer's wallet with the appropriate amount.

Function name	updateCustomer *REQUIRED QUERY (Update Operation)
Query	UPDATE Customer SET name = '\${name}', email = '\${email}' WHERE customer_id = \${customer_id}
URL	/customer/:id
Line Number	457
Description	Updates the customer's name and/or email based on the customer's id and user's input..

Before Update Customer Info Update Operation

Welcome Jordon Johnson!

BTC Balance: 3.1372016192305456

DEPOSIT

WITHDRAWAL

UPDATE INFORMATION

ITEMS BOUGHT

Full Name

Email

SUBMIT

After Update Customer Info Update Operation

Welcome Yordon Yohnson!

BTC Balance: 3.1372016192305456

DEPOSIT

WITHDRAWAL

UPDATE INFORMATION

ITEMS BOUGHT

Full Name

Yordon Yohnson

Email

SUBMIT

POST Requests

Function name	createCustomer
Query	INSERT INTO Customer(name, email) VALUES (\$1, \$2) RETURNING *
URL	/customer
Line Number	504
Description	Create a new Customer record with user specified values of name and email.

Function name	createMerchant
Query	INSERT INTO Merchant(bank_account_number, name, usd_owed) VALUES (\$1, \$2, \$3) RETURNING *
URL	/merchant
Line Number	547
Description	Create a Merchant for with specified bank account numbers, the name of the merchant, and a predetermined amount of money the Merchant is owed

Function name	createWallet
Query	INSERT INTO Wallet(btc_amount) VALUES (\$1) RETURNING *;
URL	/wallet
Line Number	487
Description	Create a wallet for a new customer with the specified value.

Function name	createOrder *REQUIRED QUERY (Insert Operation)
Query	<pre> INSERT INTO OrderDetails (customer_id, company_account_number, merchant_id, wallet_id, datetime, fee_percentage) VALUES (\$1, \$2, \$3, \$4, CURRENT_DATE, \$5) RETURNING * INSERT INTO LineItem (order_id, item_brand, item_name, item_usd_price, item_quantity) VALUES (\$1, \$2, \$3, \$4, \$5) RETURNING * INSERT INTO OnetimePurchase (order_id, conversion_rate, total_usd_price) VALUES (\$1, \$2, \$3) RETURNING * INSERT INTO Subscription (order_id, conversion_rate, charge_usd_price, billing_frequency, billing_duration) VALUES (\$1, \$2, \$3, \$4, \$5) RETURNING * SELECT * FROM Wallet WHERE wallet_id = \$1 UPDATE Wallet SET btc_amount = \$1 WHERE wallet_id = \$2 SELECT * FROM CompanyAccount WHERE account_number = \$1 UPDATE CompanyAccount SET btc_balance = \$1 WHERE account_number = \$2 SELECT * FROM Merchant WHERE merchant_id = \$1 UPDATE Merchant SET usd_owed = \$1 WHERE merchant_id = \$2 </pre>
URL	/order
Line Number	741

Description

Create an OrderDetails record and OnetimePurchase or Subscription record depending on the type of checkout, then update the customer's Wallet with the appropriate balance, then Bitlink's CompanyAccount with the transaction fee, and amount owed to the Merchant table.

Before Order Insert Operation

Note there is no Order 13 yet.

Orders

Order_id	Customer_id	Company_account_nu...	Merchant_id	Wallet_id	Datetime	Fee_percentage
4	4	1000000004	4	4	2022-08-03T07:00:00...	0.02
5	4	1000000001	3	4	2022-08-06T07:00:00...	0.02
6	1	1000000005	5	1	2022-07-31T07:00:00...	0.02
7	2	1000000005	5	2	2022-08-01T07:00:00...	0.02
8	3	1000000005	5	3	2022-08-01T07:00:00...	0.02
9	4	1000000005	5	4	2022-08-03T07:00:00...	0.02
10	5	1000000005	5	5	2022-08-06T07:00:00...	0.02
11	4	1000000001	2	1	2022-08-01T07:00:00...	0.05
12	4	1000000002	1	2	2022-08-01T07:00:00...	0.01

Cancel Order ▾

Zara

Order ID: 000000013

Date: 8/8/2022 9:49:33 PM

Item	Item Type	Price	Quantity	
MockItem1	MockItemType1	100	2	ADD TO CART

Item	Type	Quantity	Price
MockItem1	MockItemType1	2	100

Jordon Johnson

jjbeans@gmail.com

B PAY WITH BITLINK

1 BTC = 23912 USD

Order Total (USD): \$200

Order Total (BTC): 80.008364001338240215

After Order Insert Operation

Order 13 inserted at the bottom.

Orders

Order_id	Customer_id	Company_account_nu...	Merchant_id	Wallet_id	Datetime	Fee_percentage
5	4	1000000001	3	4	2022-08-06T07:00:00....	0.02
6	1	1000000005	5	1	2022-07-31T07:00:00....	0.02
7	2	1000000005	5	2	2022-08-01T07:00:00....	0.02
8	3	1000000005	5	3	2022-08-01T07:00:00....	0.02
9	4	1000000005	5	4	2022-08-03T07:00:00....	0.02
10	5	1000000005	5	5	2022-08-06T07:00:00....	0.02
11	4	1000000001	2	1	2022-08-01T07:00:00....	0.05
12	4	1000000002	1	2	2022-08-01T07:00:00....	0.01
13	4	1000000001	3	4	2022-08-08T07:00:00....	0.02

Cancel Order ▼

Function name	createDepositTransaction
Query	<pre>INSERT INTO Transaction (wallet_id) VALUES (\$1) RETURNING * INSERT INTO Deposit (transaction_id, bitlink_btc_address, btc_to_deposit) VALUES (\$1, \$2, \$3) RETURNING * SELECT * FROM Wallet WHERE wallet_id = \$1 UPDATE Wallet SET btc_amount = \$1 WHERE wallet_id = \$2</pre>
URL	/deposit
Line Number	854
Description	Create a Transaction record along with a Deposit record child, then update the customer's wallet with the amount deposited.

Function name	createWithdrawalTransaction
Query	<pre>INSERT INTO Transaction (wallet_id) VALUES (\$1) RETURNING * INSERT INTO Withdrawal (transaction_id, customer_btc_address, btc_to_withdraw) VALUES (\$1, \$2, \$3) RETURNING *</pre>

	SELECT * FROM Wallet WHERE wallet_id = \$1 UPDATE Wallet SET btc_amount = \$1 WHERE wallet_id = \$2
<i>URL</i>	/withdrawal
<i>Line Number</i>	904
<i>Description</i>	Create a Transaction record along with a Withdrawal record child, then update the customer's wallet with the amount withdrawn.

<i>Function name</i>	createItemType
<i>Query</i>	INSERT INTO LineItemType(item_name, item_type) VALUES (\$1, \$2) RETURNING *
<i>URL</i>	/itemType
<i>Line Number</i>	523
<i>Description</i>	Create an Item Type

DELETE Requests

<i>Function name</i>	deleteOrder *REQUIRED QUERY (DELETE OPERATION)
<i>Query</i>	DELETE FROM OrderDetails WHERE order_id = \$1 DELETE FROM OnetimePurchase WHERE order_id = \$1 DELETE FROM Subscription WHERE order_id = \$1
<i>URL</i>	/order/:id
<i>Line Number</i>	933
<i>Description</i>	Deletes an order from the OrderDetails table, and either OnetimePurchase or Subscription child table depending on the ISA relation. Also causes a ON DELETE CASCADE operation on the LineItem table weak entity.

Before Delete Order Delete Operation

Orders

Order_id	Customer_id	Company_account_nu...	Merchant_id	Wallet_id	Datetime	Fee_percentage
1	4	1000000001	3	4	2022-07-31T07:00:00....	0.02
2	1	1000000001	2	1	2022-08-01T07:00:00....	0.05
3	2	1000000002	1	2	2022-08-01T07:00:00....	0.01
4	4	1000000004	4	4	2022-08-03T07:00:00....	0.02
5	4	1000000001	3	4	2022-08-06T07:00:00....	0.02
6	1	1000000005	5	1	2022-07-31T07:00:00....	0.02
7	2	1000000005	5	2	2022-08-01T07:00:00....	0.02
8	3	1000000005	5	3	2022-08-01T07:00:00....	0.02
9	4	1000000005	5	4	2022-08-03T07:00:00....	0.02

Cancel Order ▾

Items Bought

YYYY-MM-DD

FILTER DATE

Order_id	Name	Datetime	Item_name	Item_usd_price	Item_quantity
1	Zara	2022-07-31T07:00:00....	Dress Shirt	30	1
4	Apple	2022-08-03T07:00:00....	Macbook	1230	1
5	Zara	2022-08-06T07:00:00....	Scarf	256	1
9	Netflix	2022-08-03T07:00:00....	Two Month Subscription	100	1
13	Zara	2022-08-08T07:00:00....	MockItem1	100	2
14	Zara	2022-08-08T07:00:00....	Cloth1	1	1
14	Zara	2022-08-08T07:00:00....	Cloth2	1	1
14	Zara	2022-08-08T07:00:00....	Cloth3	1	1
14	Zara	2022-08-08T07:00:00....	Cloth5	1	1
14	Zara	2022-08-08T07:00:00....	Cloth4	1	1

After Delete Order Delete Operation

Order 1 has been deleted using the Cancel Order drop down.

Filter Columns:

Order Number

Customer ID

Company Account Number

Merchant ID

Wallet ID

Date Time

Fee Percentage

Orders

Order_id	Customer_id	Company_account_nu...	Merchant_id	Wallet_id	Datetime	Fee_percentage
2	1	1000000001	2	1	2022-08-01T07:00:00....	0.05
3	2	1000000002	1	2	2022-08-01T07:00:00....	0.01
4	4	1000000004	4	4	2022-08-03T07:00:00....	0.02
5	4	1000000001	3	4	2022-08-06T07:00:00....	0.02
6	1	1000000005	5	1	2022-07-31T07:00:00....	0.02
7	2	1000000005	5	2	2022-08-01T07:00:00....	0.02
8	3	1000000005	5	3	2022-08-01T07:00:00....	0.02
9	4	1000000005	5	4	2022-08-03T07:00:00....	0.02
10	5	1000000005	5	5	2022-08-06T07:00:00....	0.02

Cancel Order

The LinelItem weak entity related to Order 1 has automatically been deleted on cascade.

Items Bought

YYYY-MM-DD

FILTER DATE

Order_id	Name	Datetime	Item_name	Item_usd_price	Item_quantity
4	Apple	2022-08-03T07:00:00....	Macbook	1230	1
5	Zara	2022-08-06T07:00:00....	Scarf	256	1
9	Netflix	2022-08-03T07:00:00....	Two Month Subscription	100	1
13	Zara	2022-08-08T07:00:00....	MockItem1	100	2
14	Zara	2022-08-08T07:00:00....	Cloth1	1	1
14	Zara	2022-08-08T07:00:00....	Cloth2	1	1
14	Zara	2022-08-08T07:00:00....	Cloth3	1	1
14	Zara	2022-08-08T07:00:00....	Cloth5	1	1
14	Zara	2022-08-08T07:00:00....	Cloth4	1	1