Team 3: Database Design

Simran Bawaskar, Saloni Bhutada, Aman Maheshwari,
Reha Patel, Niraj Sai Prasad, Sindhu Swaroop

DAMG 6210
Thursday Class

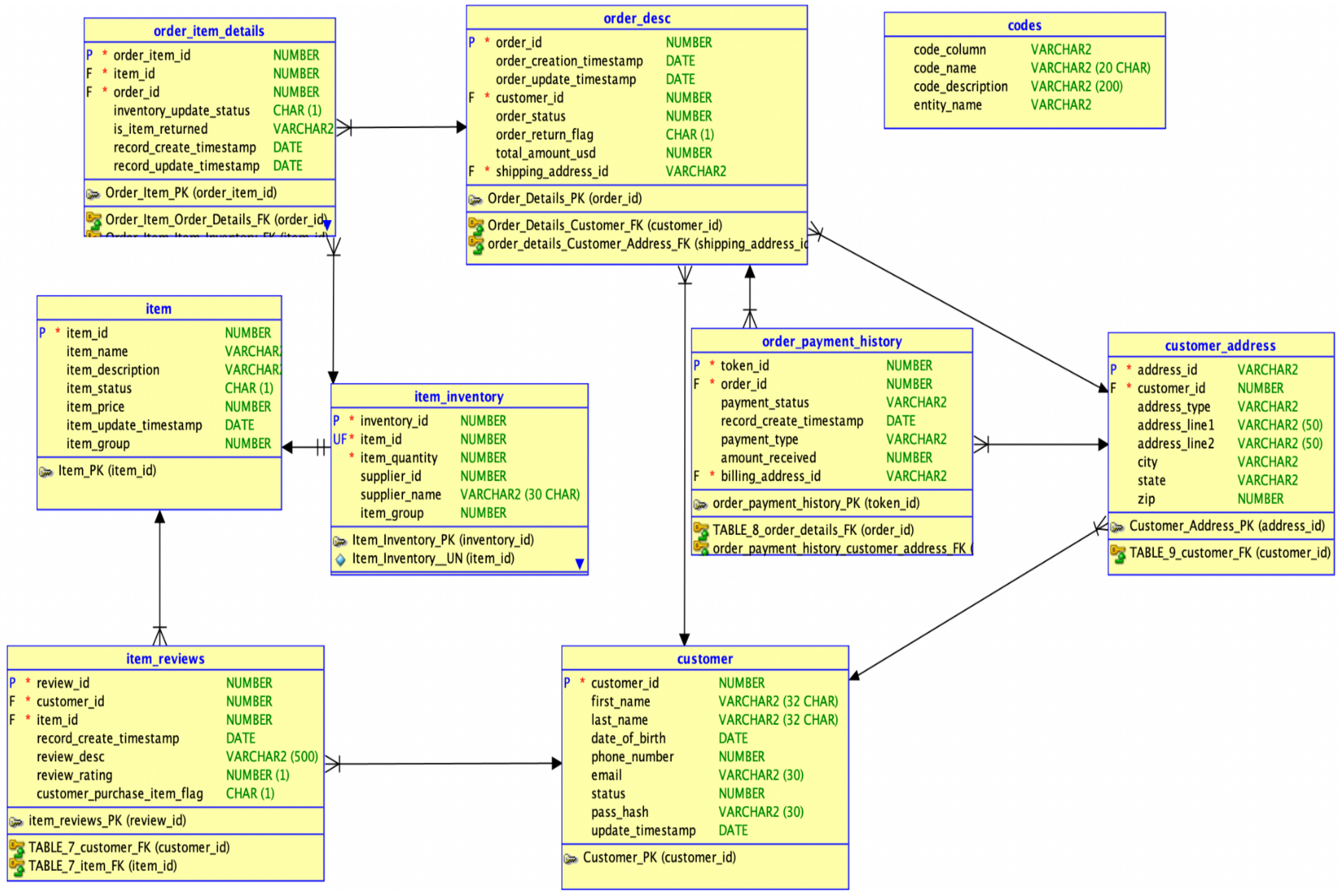
Table of Contents

1. **Business Problem & Solution**

In the retail industry, it is vital that companies are able to keep up with the supply and demand on products they are offering. Some companies may choose to keep track of this supply and demand through physical books and paperwork, and over time this can result in insufficient storage space, misplaced orders by human error, and overall customer dissatisfaction.

Our proposed solution demonstrates how a retail company that sells electronic items goes about digitally managing inventory based on orders that are fulfilled and returned. First and foremost, we allow customers to create accounts within our system that store data such as first name, last name, address, etc. This data is used when a customer creates an order with different items to ensure that the right orders are being associated with the right customers. In addition to this, when an order is placed, we will make sure that the correct amount of each item in the order is removed from the inventory. On the other hand, if a customer wants to return an item, we also have the capability to create a return order and readjust the inventory of the item.

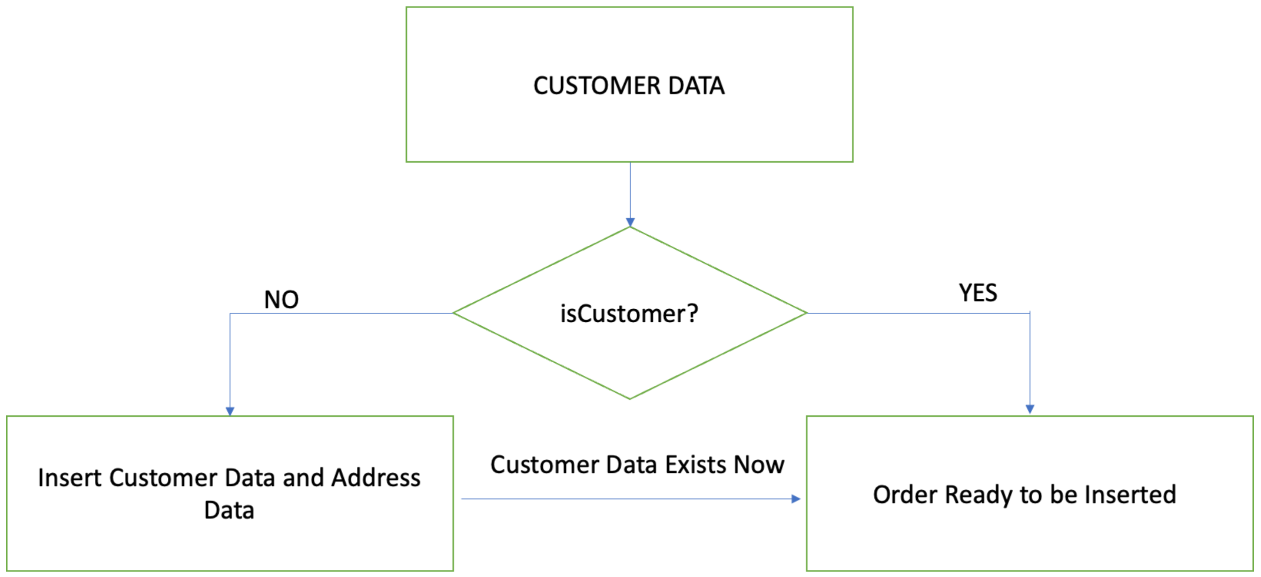
2. Database Model


order_item_details
P * order_item_id NUMBER
F * item_id NUMBER
F * order_id NUMBER
inventory_update_status CHAR (1)
is_item_returned VARCHAR2
record_create_timestamp DATE
record_update_timestamp DATE
Order_Item_PK (order_item_id)
Order_Item_Order_Details_FK (order_id)
order_desc
P * order_id NUMBER
order_creation_timestamp DATE
order_update_timestamp DATE
F * customer_id NUMBER
order_status NUMBER
order_return_flag CHAR (1)
total_amount_usd NUMBER
F * shipping_address_id VARCHAR2
Order_Details_PK (order_id)
Order_Details_Customer_FK (customer_id)
order_details_Customer_Address_FK (shipping_address_id
codes
code_column VARCHAR2
code_name VARCHAR2 (20 CHAR)
code_description VARCHAR2 (200)
entity_name VARCHAR2
item
P * item_id NUMBER
item_name VARCHAR2
item_description VARCHAR2
item_status CHAR (1)
item_price NUMBER
item_update_timestamp DATE
item_group NUMBER
Item_PK (item_id)
item_inventory
P * inventory_id NUMBER
UF * item_id NUMBER
* item_quantity NUMBER
supplier_id NUMBER
supplier_name VARCHAR2 (30 CHAR)
item_group NUMBER
Item_Inventory_PK (inventory_id)
Item_Inventory__UN (item_id)
order_payment_history
P * token_id NUMBER
F * order_id NUMBER
payment_status VARCHAR2
record_create_timestamp DATE
payment_type VARCHAR2
amount_received NUMBER
F * billing_address_id VARCHAR2
order_payment_history_PK (token_id)
TABLE_8_order_details_FK (order_id)
order_payment_history_customer_address_FK
customer_address
P * address_id VARCHAR2
F * customer_id NUMBER
address_type VARCHAR2
address_line1 VARCHAR2 (50)
address_line2 VARCHAR2 (50)
city VARCHAR2
state VARCHAR2
zip NUMBER
Customer_Address_PK (address_id)
TABLE_9_customer_FK (customer_id)
item_reviews
P * review_id NUMBER
F * customer_id NUMBER
F * item_id NUMBER
record_create_timestamp DATE
review_desc VARCHAR2 (500)
review_rating NUMBER (1)
customer_purchase_item_flag CHAR (1)
item_reviews_PK (review_id)
TABLE_7_customer_FK (customer_id)
TABLE_7_item_FK (item_id)
customer
P * customer_id NUMBER
first_name VARCHAR2 (32 CHAR)
last_name VARCHAR2 (32 CHAR)
date_of_birth DATE
phone_number NUMBER
email VARCHAR2 (30)
status NUMBER
pass_hash VARCHAR2 (30)
update_timestamp DATE
Customer_PK (customer_id)

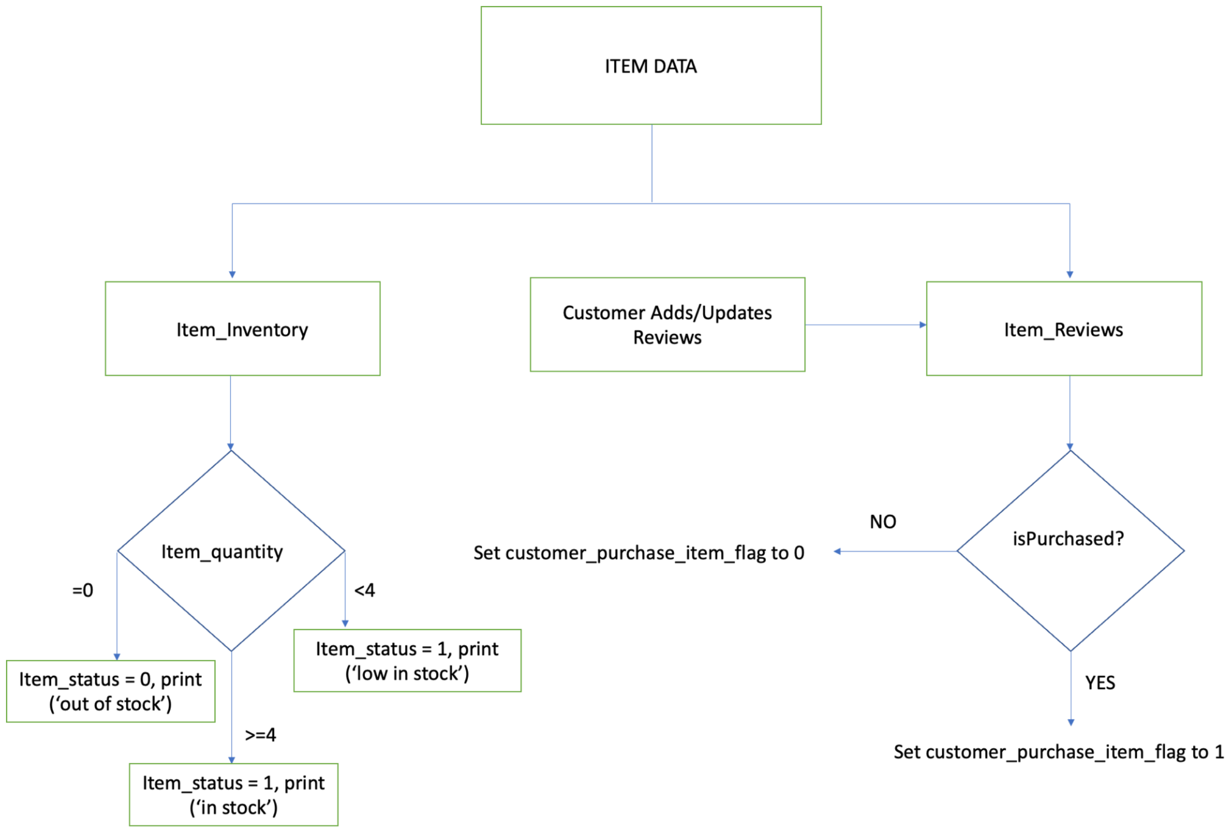
Our database design consists of 9 tables:

- **Customer table** - Consists of customer demographics and login information
- **Customer_Address table** - Consists of billing/shipping addresses
- **Order_Details table** - Consists of specific details of orders made by the customers
- **Order_Item table** - Consists of status of item inventory and relationships between orders and items
- **Item_Inventory table** - Consists of item quantities
- **Item table** - Consists of item information
- **Item_Reviews table -** Consists of all customer reviews of a particular table
- **Order_Payment_History table -** Consists of all payments paid against the order
- **Codes table -** A legend to all codes used in various tables in our database

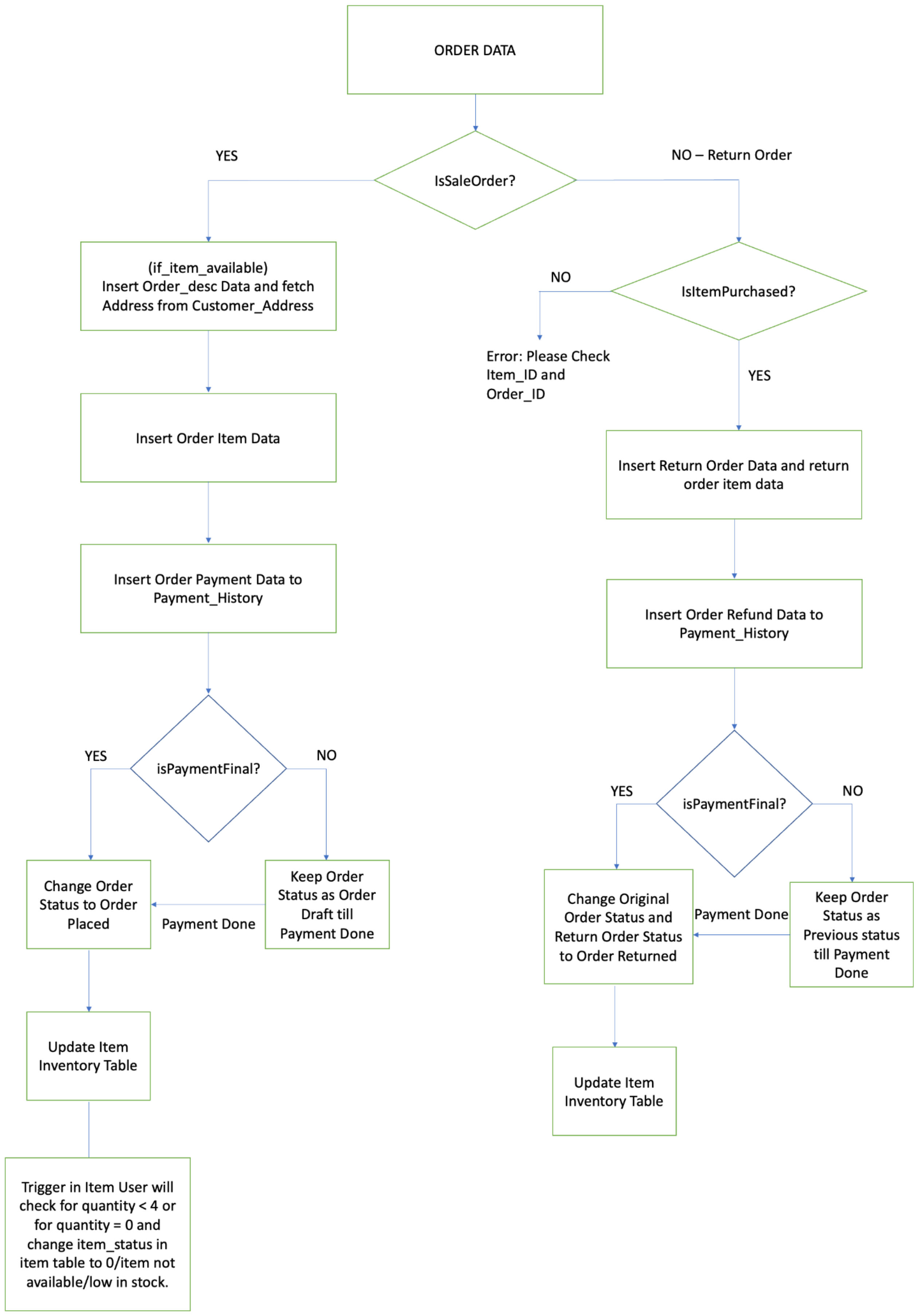
3. Data Flow Diagrams



CUSTOMER DATA
isCustomer?
NO
YES
Insert Customer Data and Address Data
Customer Data Exists Now
Order Ready to be Inserted



ITEM DATA
Item_Inventory
Customer Adds/Updates Reviews
Item_Reviews
Item_quantity
=0
<4
>=4
Item_status = 0, print ('out of stock')
Item_status = 1, print ('low in stock')
Item_status = 1, print ('in stock')
isPurchased?
NO
Set customer_purchase_item_flag to 0
YES
Set customer_purchase_item_flag to 1


ORDER DATA
IsSaleOrder?
YES
NO – Return Order
(if_item_available) Insert Order_desc Data and fetch Address from Customer_Address
Insert Order Item Data
Insert Order Payment Data to Payment_History
isPaymentFinal?
YES
NO
Change Order Status to Order Placed
Payment Done
Keep Order Status as Order Draft till Payment Done
Update Item Inventory Table
Trigger in Item User will check for quantity < 4 or for quantity = 0 and change item_status in item table to 0/item not available/low in stock.
IsItemPurchased?
NO
Error: Please Check Item_ID and Order_ID
YES
Insert Return Order Data and return order item data
Insert Order Refund Data to Payment_History
isPaymentFinal?
YES
NO
Change Original Order Status and Return Order Status to Order Returned
Payment Done
Keep Order Status as Previous status till Payment Done
Update Item Inventory Table

4. Business Rules

1) Our company wants to focus only on Electronic Items as a main product. Our inventory will include items like cell phones, cell phone accessories, laptops, etc.
2) In our project, shipping_address_id will point to the address_id column in the table Customer_Address. In the Customer_Address table, an address may be just a shipping address, just a billing address, or both. Each of these categories will be given a code accordingly.
3) Our company is relatively new and therefore we only have one supplier for each item at the moment. As the company continues to grow and expand, we may add more suppliers in the future.
4) If a customer wants to return an order then the return request will be considered as a new return order. The new return order will behave similar to a new order request, except the amount paid will be refunded and the item_quantity will be adjusted.
5) The Codes reference table will be pre-set for all the tables that have a status column.
6) A customer may deactivate their profile, but their tuple will not be removed from the customer table. Instead, their status will change from 1 (active) to 0 (inactive).
7) Our company does not accept cash payments - it only accepts debit, credit, and gift cards. We know a payment has been registered once the record is added to the Order_Payment_History table.
8) When a customer is trying to return an item purchased in a sale order, our system first checks if the customer has actually purchased the item in the said order. If everything returns TRUE, the order is return-eligible.

5. Views

Throughout our database we will be implementing views to facilitate our company's order management and fulfillment operations. The views below are integral for the process because we do not want to always display information such as update timestamps, statuses, passwords, etc. and views will help us decide which information is visible to which roles. The following are views we will be implementing:

- **Customer_View:** Consists of the following columns: customer.first_name, customer.last_name, customer_address.address_line1, customer_address.address_line2, customer_address.city, customer_address.zip, customer_address.state, customer.phone_number, customer.email will display's customers demographic information that can be accessed and modified by the customer.
- **Item_View:** Consists of the following columns: item.item_name, item.item_description, item.item_price, and item_inventory.item_quantity for customers to access the items and their details.
- **Order_Desc_Item_Payment_View:** This view will consist of three tables: Order_Item_Details, Order_Desc, and Order_Payment_history. It will consist of the following columns: order_desc.order_id, order_payment_history.payment_status, order_desc.order_status, order_desc.total_amount_usd, order_payment_history.payment_type, order_item_details.item_id to help the customers to keep track of every transaction.
- **Item_Reviews_View:** Consists of the following columns: item_reviews.item_id, item.item_name, item.item_description, item_reviews.review_desc, item_reviews.customer_id, customer.first_name, item_reviews.review_rating for customers to access the reviews of different items.
- **Order_Desc_Item_Customer_View:** This view will consist of three tables: Order_Item_Details, Order_Desc, and Customer. It will have the following columns: order_desc.order_id, order_desc.customer_id, customer.first_name, customer.last_name, order_desc.order_status, order_desc.total_amount_usd, and order_item_details.item_id, item.item_name.

6. Security Details

Team Member Roles

1. **Database Administrator - Simran:** The database administrator (DBA) will be responsible for monitoring and assisting the database in all stages from development to install. The DBA should be able to assist the team when the database connection lags or when the overall performance of the database is down. The DBA will have access to CREATE, UPDATE, and DELETE.
2. **Application DBA - Reha, Aman:** The application database administrator (DBA) will be responsible for creating the users within the database as well as determining which users have access to specific database objects and actions. Alongside this, the application DBA is responsible for granting roles and privileges when the developers need access. The Application DBA will have access to CREATE, UPDATE, and DELETE. Additionally, the Application DBA will be responsible for creating the Codes table which is referenced throughout the database.
3. **Monitoring User - Sindhu, Saloni, Niraj:** The monitoring user will be responsible for checking for performance issues within the database as well as performing metrics on the different procedures in the database. Such performance will help ensure that the database is performing well, and if there are any issues, the monitoring user should work with the DBA to fix the problem in a timely manner.
4. **Application Developer - All Members:** The application developers will break down the business requirements and propose possible solutions to the team. They will be utilizing PL/SQL to create tables, views, procedures, etc. that will satisfy the problem at hand in an efficient way. Alongside this, they are responsible for testing their code alongside the Application DBA to ensure that they have the appropriate roles required. Developers will be given privileges to CREATE and UPDATE tables, but they will need to work with a DBA to DELETE a table to ensure that no sensitive information is being deleted.

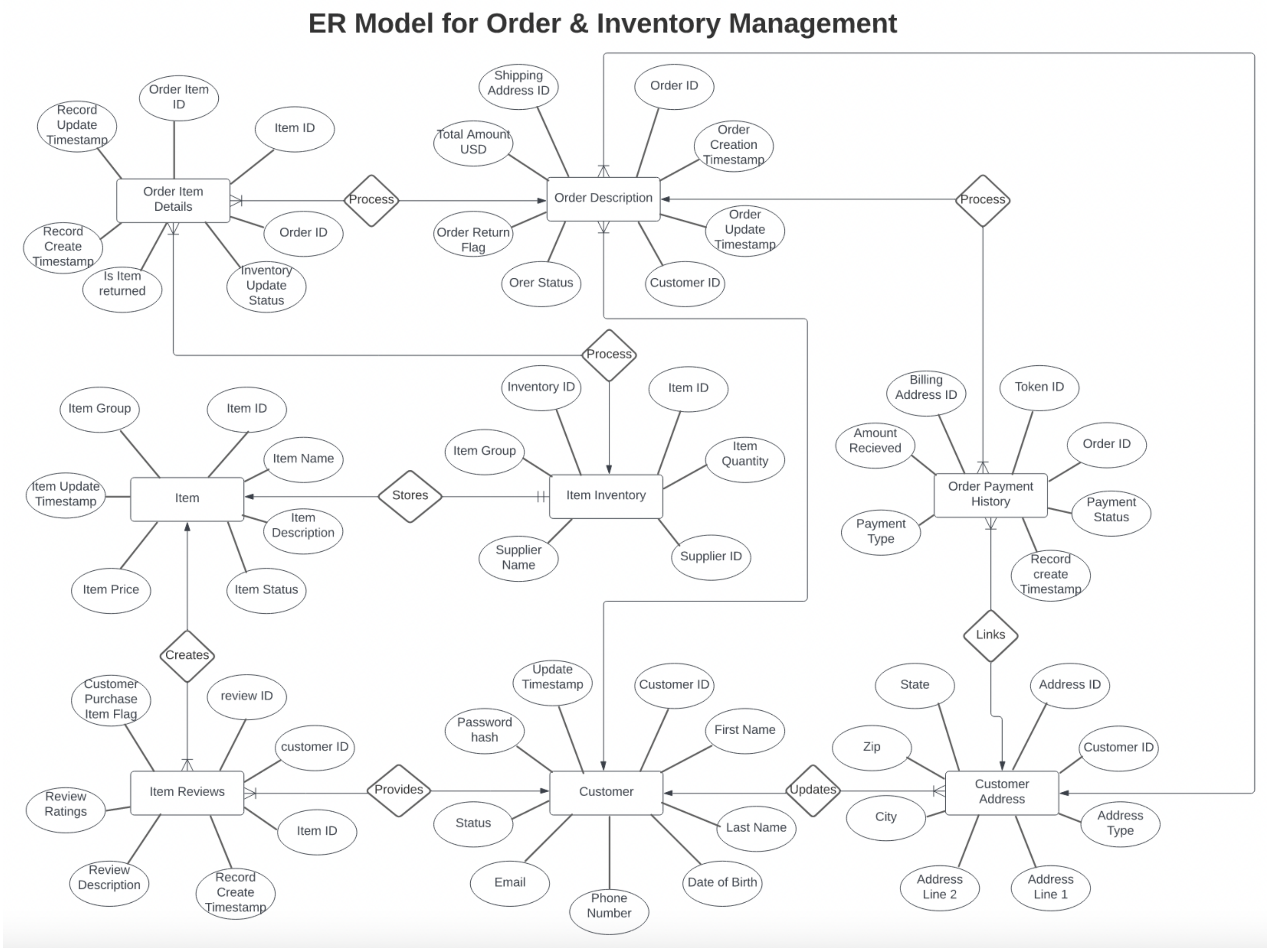
User Roles

1. **Customer_Owner:** The Customer_Owner role is responsible for creating and maintaining the tables called Customer and Customer_Address. They will be able to CREATE and UPDATE these tables and with the assistance of a DBA, they will also be able to DELETE. They will also have UPDATE privileges for Item_Reviews, Order_Payment_History, Order_Item_Details and Order_ Desc. Additionally, they will have READ access to Item_View, Order_Desc_Item_Payment_View, and Item_Reviews_View.
2. **Order_Owner:** The Order_Owner role is responsible for creating and maintaining the tables called Order_Desc, Order_Payment_History, and Order_Item Details. Similar to the previous role, they will be able to CREATE and UPDATE the previously mentioned tables and with DBA assistance they will be able to DELETE as well. They will also

have UPDATE privileges to Item_Inventory. Finally, they will have READ access to Item_View, Order_Desc_Item_Payment_View, and Customer_Address.

3. **Item_Owner:** The Item_Owner role is responsible for creating and maintaining the tables called Item, Item_Inventory, and Item_Reviews. Similar to the previous two roles, they will be able to CREATE and UPDATE the previously mentioned tables and with DBA assistance they will be able to DELETE as well. They will have READ access to Customer_View and Order_Desc_Item_Customer_View.

7. ER Diagram



ER Model for Order & Inventory Management
Order Item Details
Record Update Timestamp
Order Item ID
Item ID
Record Create Timestamp
Order ID
Is Item returned
Inventory Update Status
Process
Order Description
Shipping Address ID
Total Amount USD
Order ID
Order Creation Timestamp
Order Return Flag
Order Update Timestamp
Orer Status
Customer ID
Process
Process
Item
Item Group
Item ID
Item Name
Item Update Timestamp
Item Description
Item Price
Item Status
Stores
Item Inventory
Inventory ID
Item ID
Item Group
Item Quantity
Supplier Name
Supplier ID
Order Payment History
Billing Address ID
Token ID
Amount Recieved
Order ID
Payment Status
Payment Type
Record create Timestamp
Links
Creates
Item Reviews
Customer Purchase Item Flag
review ID
customer ID
Review Ratings
Item ID
Review Description
Record Create Timestamp
Provides
Customer
Update Timestamp
Customer ID
Password hash
First Name
Status
Last Name
Email
Phone Number
Date of Birth
Updates
Customer Address
State
Address ID
Zip
Customer ID
City
Address Type
Address Line 2
Address Line 1