**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

INITECH. Inc. 

Project: Archbike Database

Report By: Dalajid Dizdarevic

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Date: 05/02/2023

Subject: This report will cover the status of the ArchBike database and give a detailed explanation of the structural foundation and choices behind its creation.

Status: This final report serves to document and explain the ArchBike Database in its entirety. There are multiple sections for this report, and they are listed as such: Features, User Interface, User Help, Data Dictionary, and Program Documentation. Features will give an explanation of all the unique features provided by the database application. User Interface will show sample screens from the application and explain the structure of the database. User Help will give an explanation and document all of the various screens that are implemented to help users. Data Dictionary is a listing of every single table within the database and explains its contents in depth as well as shows the diagrams made in its creation.. Program Documentation is the last section, which is a section that contains all of the code, diagrams, and other technical aspects of this application. With this report, we believe we have created a comprehensive and user friendly database that will serve all of ArchBike’s needs.

Features:

This application provides many different features and benefits to ArchBike employees. Some of the more important benefits include:

Home Page/Menu - This feature offers easy navigation between reports and/or pages that would be required for use by ArchBike employees. Since all the different pages are listed as cards/buttons on the home page, there is no typing required until form entry, to try and make the process as easy as pointing and clicking where the user wants to go.   
  
Graphs - This feature is a graph that shows sales and various other metrics that are already calculated by ArchBike managerial staff when used to see monthly/weekly reports. These are important figures to be calculated, so adding these into the program creates the added benefit of automating that system for ArchBike and localizing it all within the same program.

Contact/Support - This feature offers a direct method of communication with our development team that will allow for quick and easy communication between users and developers. This is important so that if any questions, comments, or suggestions must be made from the users, then they have a direct hotline to send it to that can be quickly sorted by our team and handled accordingly.

Search - This feature will allow users to search for different entries within the database and allow them to either search for specific entries or entries that share a common attribute. This process is streamlined as well, as it also abides by the same “point and click” methodology of the home screen, so if a user wants to add specifications to a search, then all they have to do is click those specifications that are already listed on the screen.

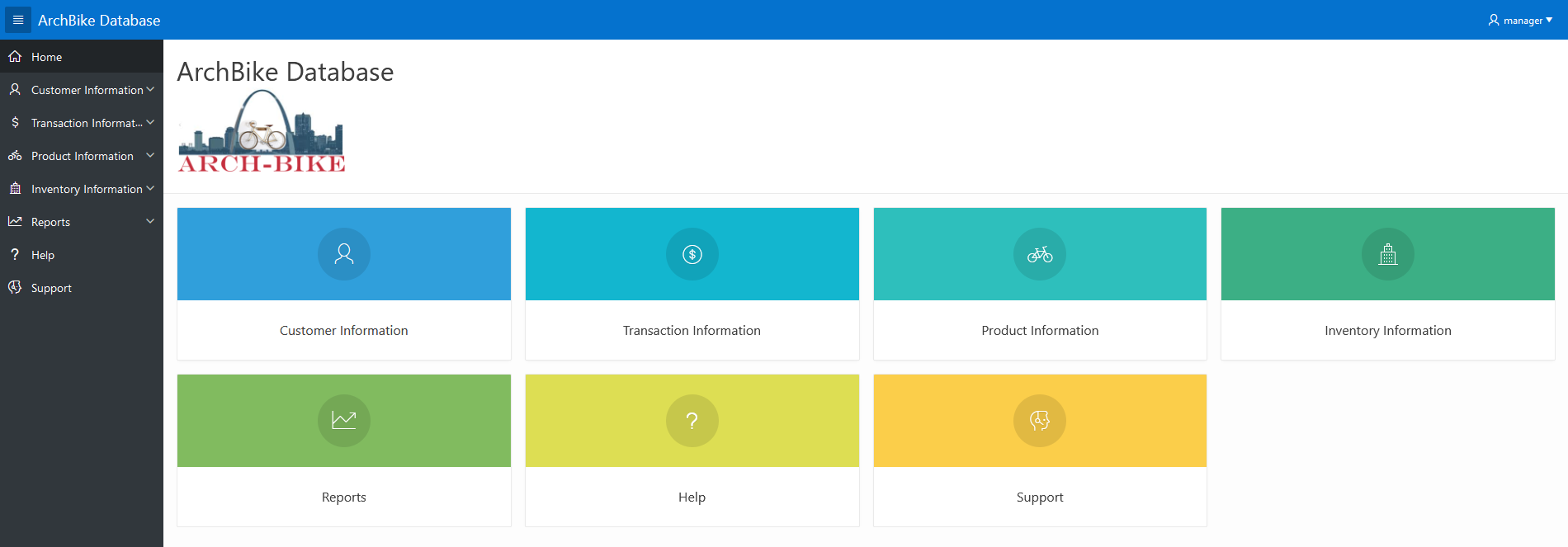
User Access Control - This feature is more “backend” focused in the sense that it is important for controlling which users can access which pages. Through the use of various user groups and authorization schemes, this application supports multiple layers of security within itself. As of now there are currently three levels, however this can easily be changed at any point during ArchBike’s operations to include more tiers to the security. Firstly is the Viewer clearance level, which can be given to employees who do not have a need for editing the database, so they can only view all non-managerial pages. Next is the Editing clearance level, which can be given to employees who are responsible for editing information within the database and adding new entries to the tables. This clearance level, similarly to the Viewer, cannot see managerial pages. The final level is the Managerial level, which has all of the benefits of the prior clearance levels with the added benefit of being able to see reports and graphs that relate to ArchBike’s sales and transactions. This sort of tiered security creates a safe environment where those who should not have access to important data, will not.

This application also generally follows a heavy “quality of life” design philosophy. It’s made in a way so that users can easily do all of the things they want to, within their security clearance.

User Interface:

This database application has been streamlined to make processes as easy as pointing and clicking where the user wants to go, at least as much as possible. To operate this application, initially users will be prompted to log into the database. From here, they can use their login credentials given to them by ArchBike, which will then allow them to access the database according to their clearance level. Everyone who has access to the database has, at the least, viewing access, so when they log in everyone will be sent to the home page, where they see the same set of pages and buttons they can click on.

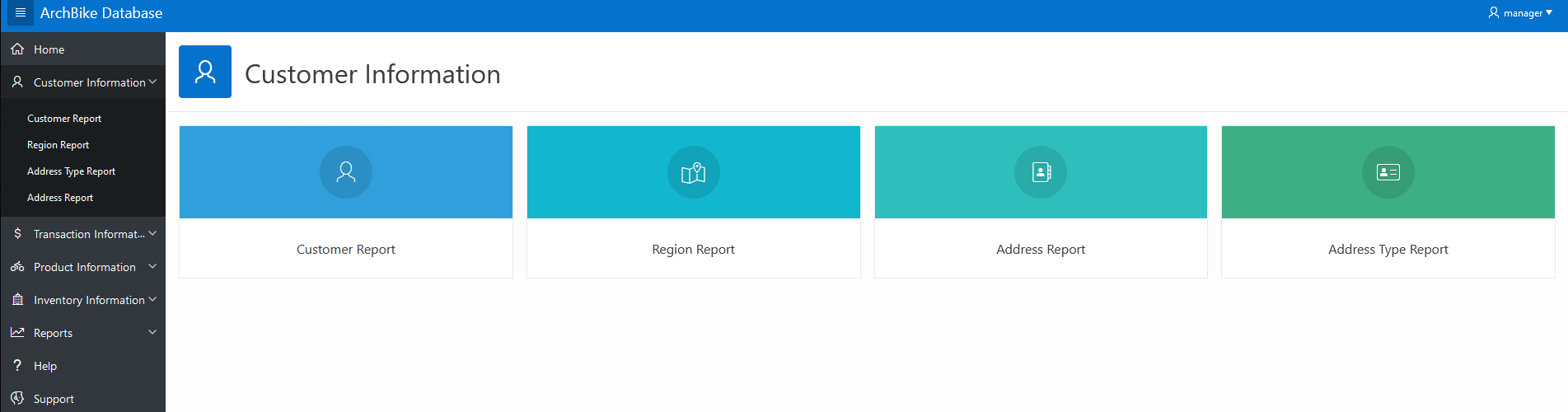
**Figure 1**  
Home Page



(*Note.* This is the central hub area for all the buttons. The view shown above is one with the managerial clearance level.)

To start traversing the database, users can click on the various sections labeled “\_\_\_ Information”, which will then take them to that respective section of the database. For a more comprehensive guide of which section contains which pages of the database, a help section has also been implemented in the home page as well. After clicking on an “\_\_\_ Information” button, the user will be taken to another menu similar to the first that houses each specific report of the database.

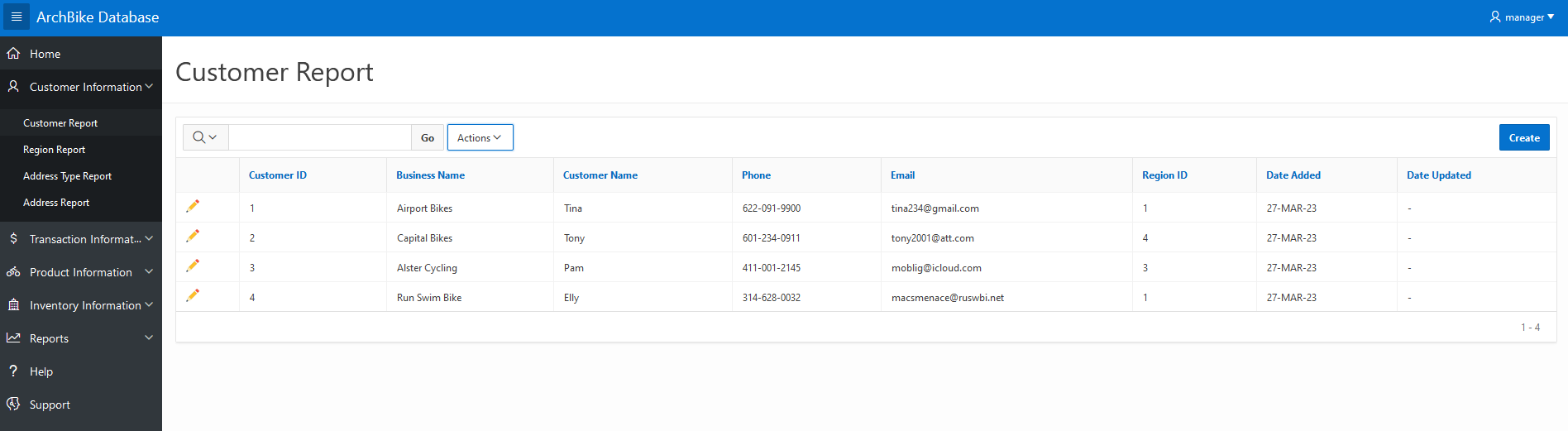
**Figure 2**  
Customer Information

  
(*Note.* These specific tables are only accessible in the customer information section. The product, transaction, and inventory sections all house different tables from this one.)

From there, the user can click on those reports to see that entire table of the database and, if given the clearance, edit those tables or add to them. Once in a report, viewing is quite simple and has multiple features to it that help organize or search the data in a specific way. These can be further explained within the application, by clicking on the “Actions” button and then clicking “Help” while in a report page.

**Figure 3**

Customer Report



For those with editing access, a create button will also be viewable that users can click to access the entry forms for these reports. This will bring up an empty form, where users will be prompted to enter certain bits of information depending on what is asked. There are multiple levels of input validation for these, which generally just follow basic logic principles (For example, you cannot enter a product’s ending date as before its start date, since the product can’t end production before its even began, or else you’ll be prompted with an error).

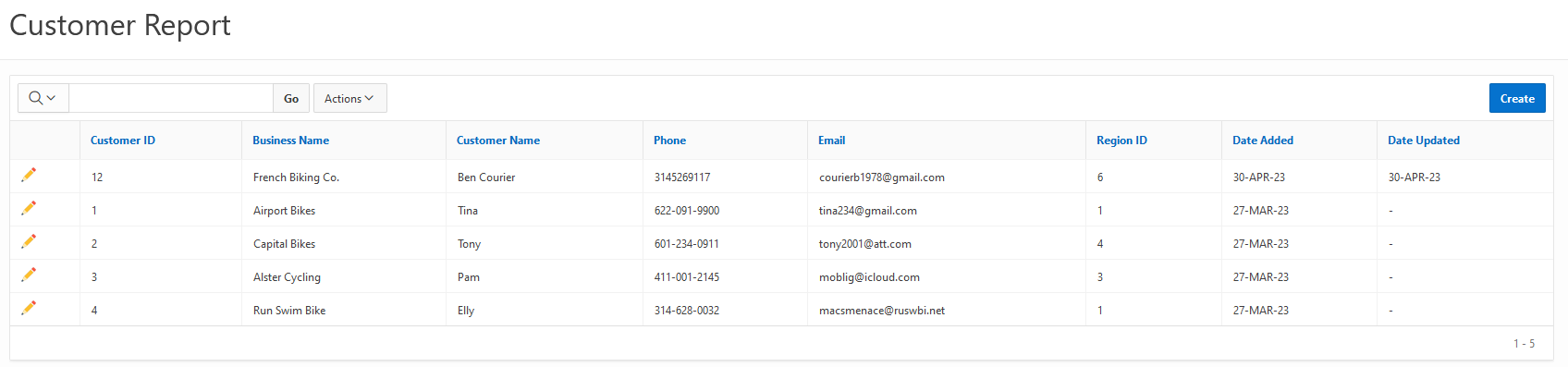
**Figure 4**

(*Note*. This figure uses an incorrect date combination as explained above, as the date a product was updated cannot possibly be before it was added to the database.)

Once a user finishes filling the fields in, they can then hit the create button which will add that new entry to the table, assuming that all the information has been entered with the proper specifications in mind.

**Figure 5**

Updated Customer Report



If a user wishes to edit or delete an entry in the report, they can click the pencil icon on the left hand side of the entry, which will then take them to the same entry form listed earlier, which will just be occupied with the data of that entry.

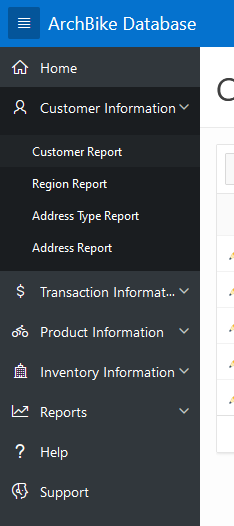
**Figure 6**

Edit Form



The user can then edit those entries listed according to the same logic principles set for the entry creation. If the user wishes to confirm the changes, they can then click the “Apply Changes'' button to submit it and the entry will be updated. If they wish to delete that entry, then they can click the “Delete” button and the entry will be removed from the table. Now if a user wants to reset back to the front page, all they have to do is click the “Home” button on the left hand side of the screen, which is located in the side menu. This menu can be used to traverse the entire database as well, and can be toggled to fully show text or hide it with the button that shows four lines in the top left corner of the screen. Drop down arrows are also available that will expand each section, which then allows users to access the tables from that side menu easily.

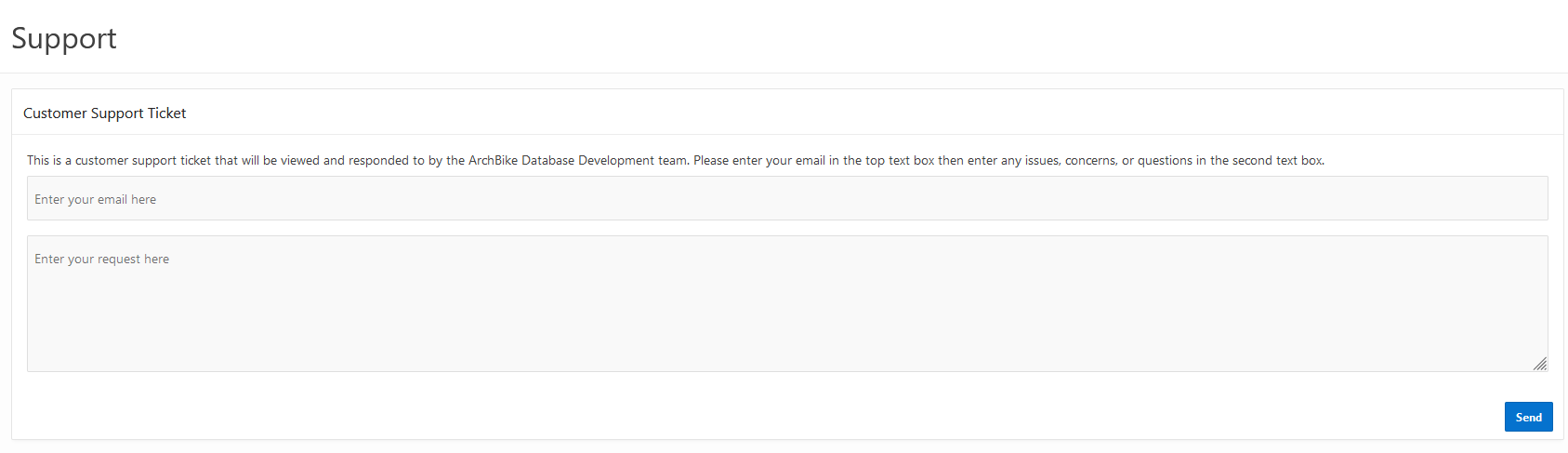
**Figure 7**  
Side Navigation Menu



The final section to explain for employee level clearance is the support button. This will send users to a customer support page that has two text boxes. The first requires the user to enter their email address, and the second is a larger text box that the user can use to list any complaints, bugs, questions, etc. that came up while using the database. Once the user fills both the boxes and hits send, they will then be sent to a thank you page, where they can then navigate back to the home menu.

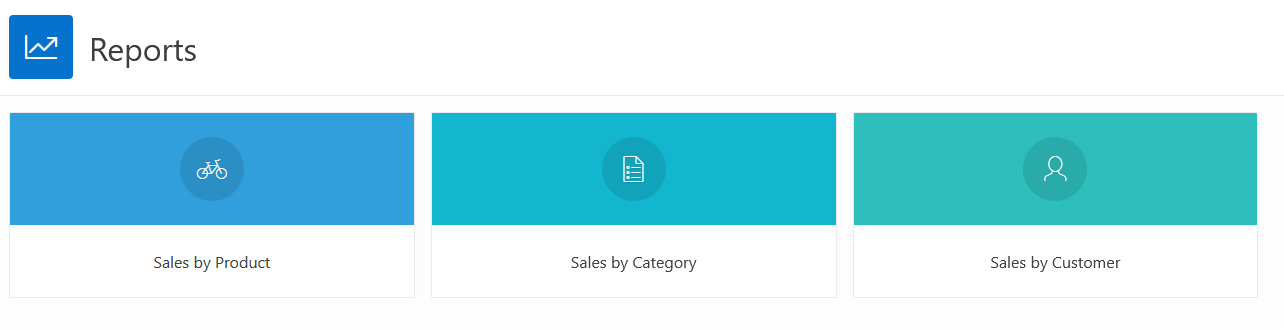
**Figure 8**

Support Ticket

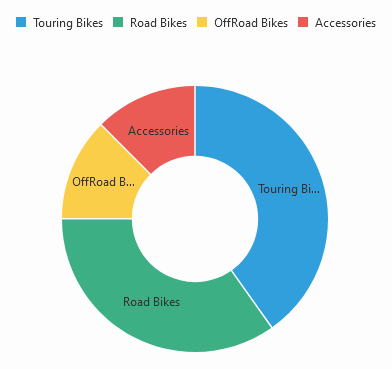


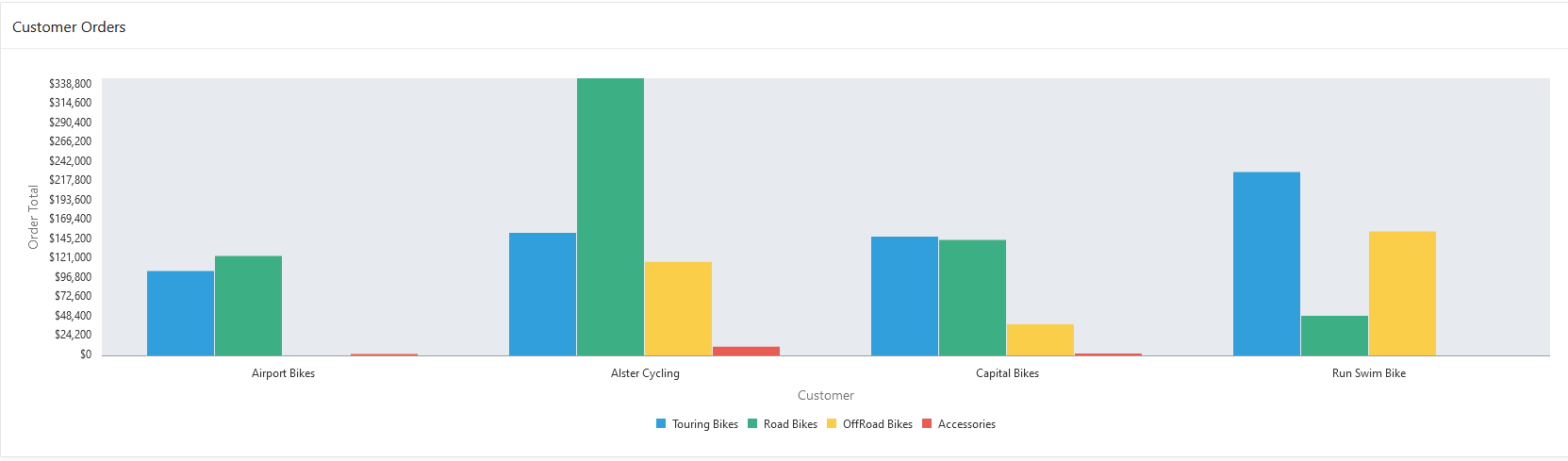
After clicking the send button, an email ticket will automatically be sent to the ArchBike development team’s email address. This email address will be used solely for the purpose of internal affairs with the tickets and will be checked daily and addressed as quickly and efficiently as possible. The last section of this database is hidden to all users except for those with managerial clearance, and this section is titled “Reports”. This section functions similarly to all the previous ones, and will take users to a page where they will see multiple report options to click on and view. There are currently three reports implemented, which are Sales by Category, Sales by Product, and Sales by Customer. These all show the sales numbers by these various metrics and show which products sell the most, the least, who buys them, etc.

**Figure 9**  
Reports







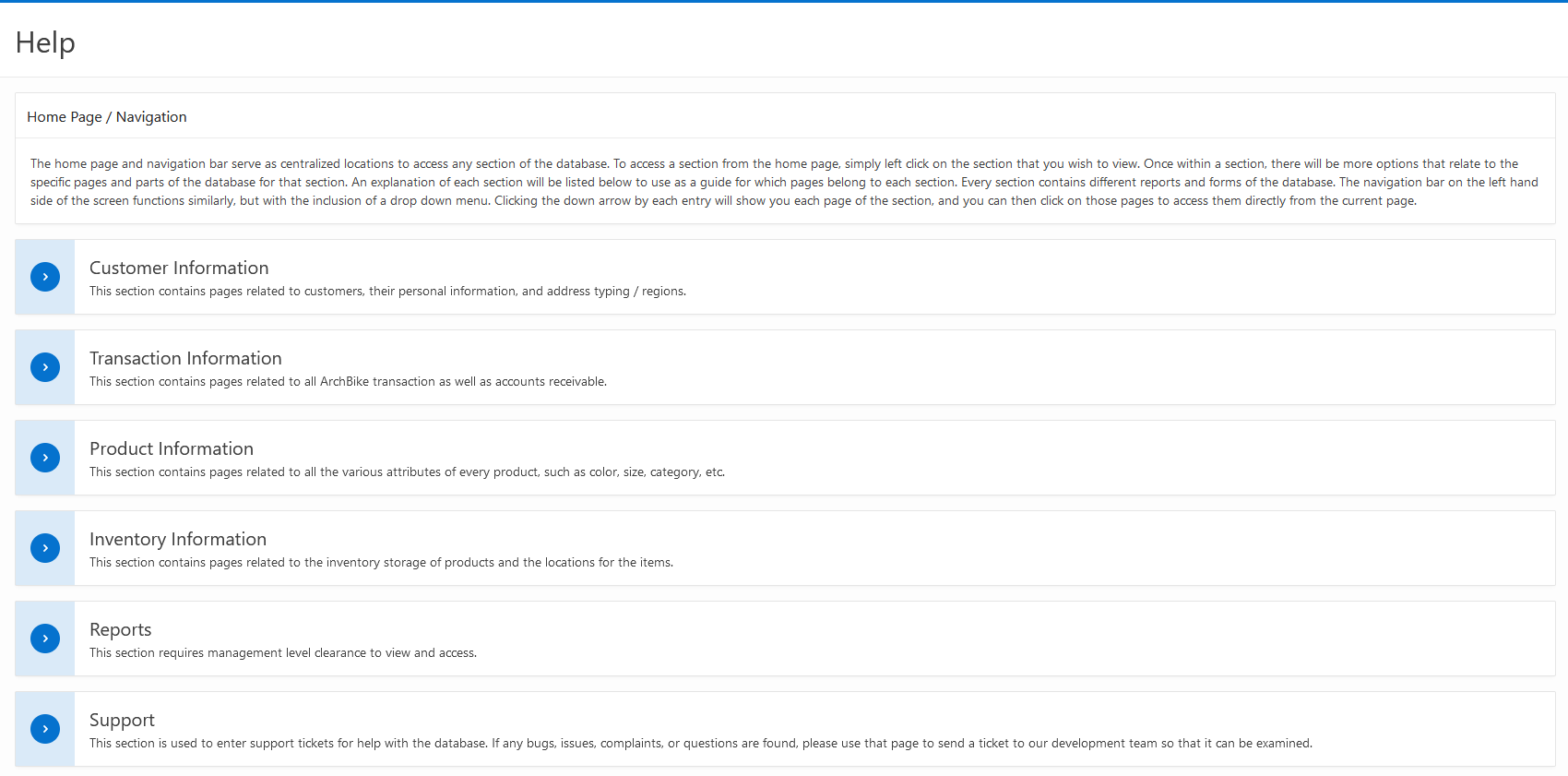


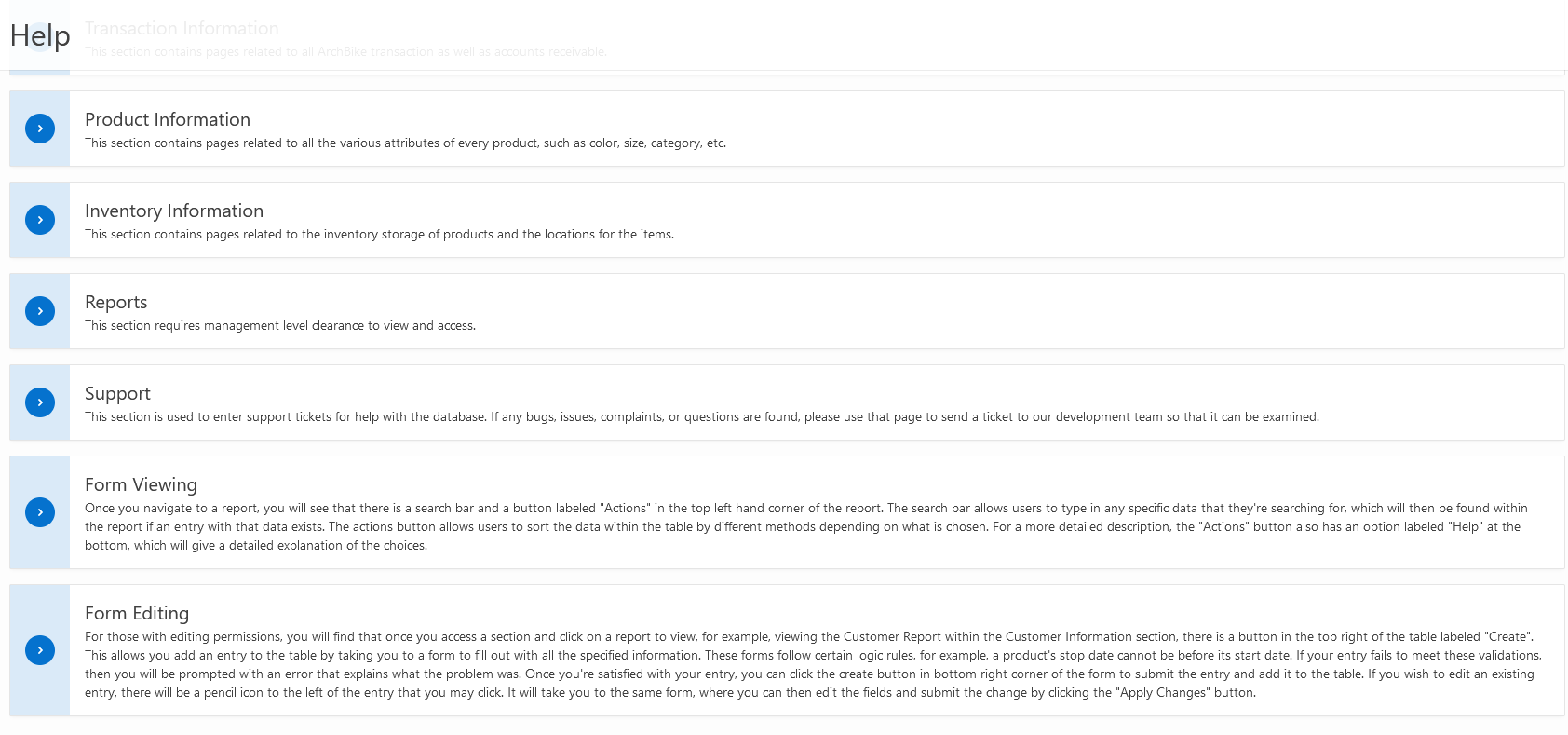
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

User Help:

There are various help screens and error messages that this application offers to try and help users that are experiencing difficulties with the program. For starters, the main help that users will see when initially opening the database is the “Help” section on the home page. This section explains the navigational system, the layout of the pages, and how to use the forms and reports within the database.   
  
**Figure 1**

Main Help Screen

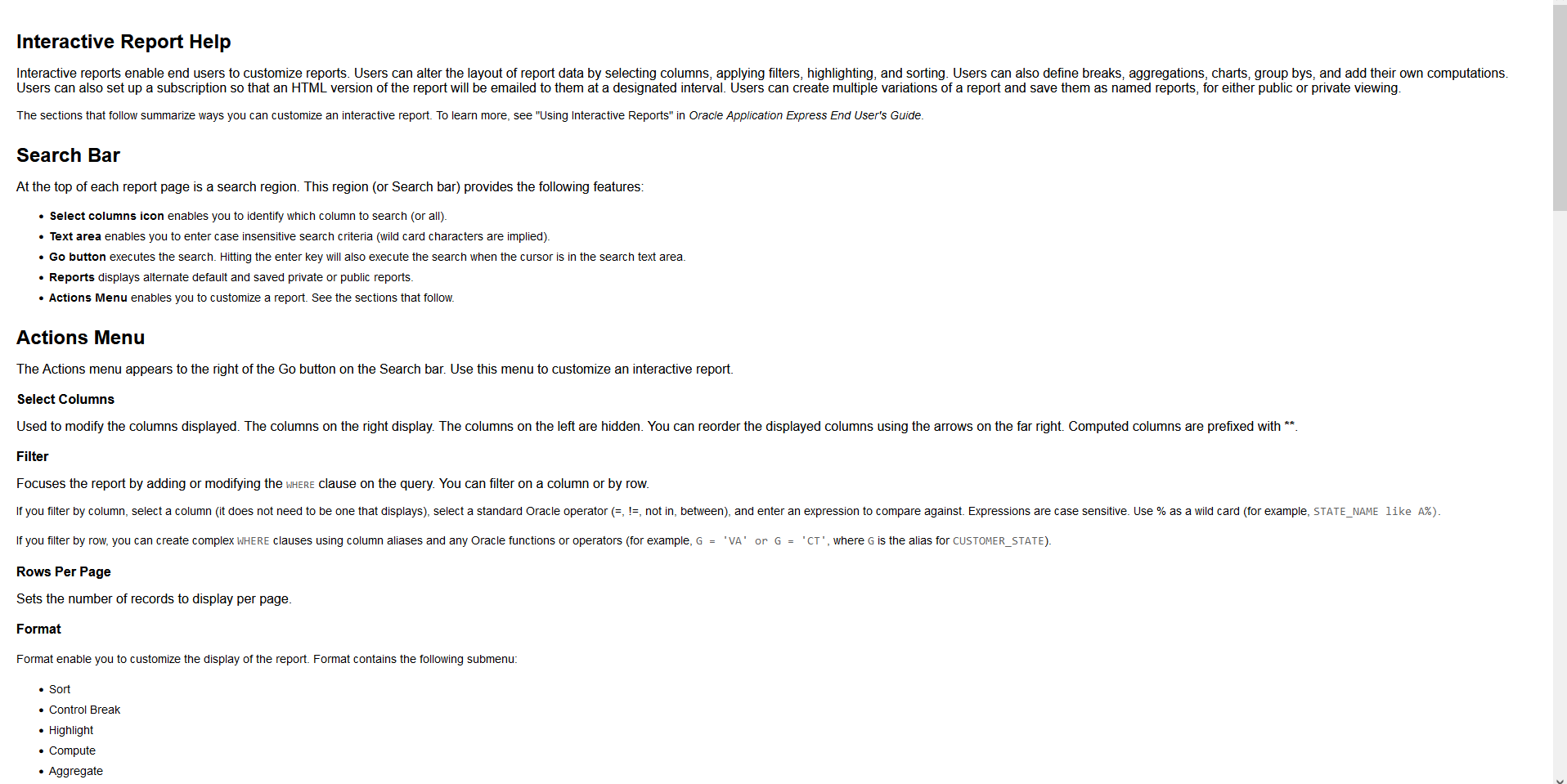




The next major help section appears when viewing reports. Users can click the “Action” button which then brings up a drop down menu, and at the bottom there is another button labeled “Help”. Clicking this will bring up another window that fully explains the viewing functionalities of the tables within the database.

**Figure 11**

Form Help Screen



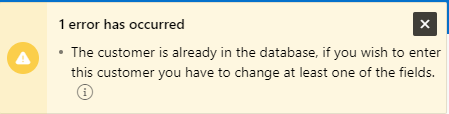
(*Note.* There is a substantial amount of information here, so for brevity sake the beginning was picked for the figure above.)

Another main form of help is the user support ticket system that has been added to the application (Shown in Figure 8). This system allows for users to send in questions, comments, concerns, bugs, etc. that they have found and also allows for our developer team to communicate with the users effectively. This helps solve any issues as they come up in the most timely manner possible. As of now, the support email listed is [ddizdarevic@webster.edu](mailto:ddizdarevic@webster.edu), so all tickets sent from that system are sent to that email and will be resolved as quickly as possible.

As for error messages, the system provides multiple forms of input validation for forms, so the majority of them come from the forms. Not all forms share the same amount of input fields, so there’s not a need for all forms to have the same amount of input validations. Every single form has validation for unique entries, meaning that if a user tries to edit or create a form with the same exact information that already exists in the database and submit that, then it will not go through and the user will be told that they cannot enter an entry that already exists.

**Figure 12**

Unique Entry Validation

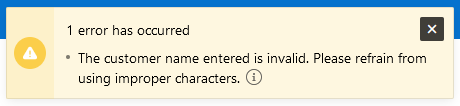


Another error message is used for when special characters are used in incorrect fields.

This appears for most text fields, but there are a few larger ones that allow for special character entry. Typically this is used to limit characters that shouldnt belong for things like names, codes, etc.

**Figure 13**

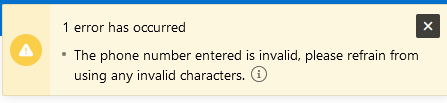
Character Validation



There’s also input validation for some fields that can only take numbers and specific special characters. This appears in fields like phone numbers, where the user needs to enter only digits, but the numbers can also contain hyphens between them.

**Figure 14**

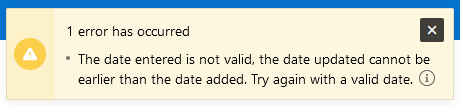
Number Validation



The last major error message that shows up for many of the forms is a form of date validation. This appears whenever a date entered cannot logically exist. For example, if a product were to start production on April 26th, 2023, then the the end date of that product’s production cannot be any date before the start date. This error message is one of the most common in the forms, as many of them require dates.

**Figure 15**

Date Validation



- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

Data Dictionary:

The following section will list out all tables in a data dictionary format to fully show each attribute, their contents, their qualities, what variables they will use, etc. for the sake of descriptive clarity and documentation.

**NAME:** Address  
**DESCRIPTION:** This table contains data about a customer’s address or addresses, which is necessary in this database. Customers can have multiple different addresses for different needs, such as billing and shipping addresses, which need to be separately stored. Different regions need to be set for each address as well to ensure company capabilities on the logistics end.

**RELATIONAL FORMAT:** Address( AddressID, Street\_Address, City, State, Zip\_Code, Country, CustomerID, TypeID )

**OTHER TABLES REFERENCED:** Customer, AddressType

| Attribute Name | Contents | Type | Size | Format | Range | Req | Key | Restrictions |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| AddressID | Identity number for address | NUMBER | Sys Size | 9(-10^38 +1 to 10^38 –1) | 0 - 999… | Yes | PK | Unique |
| Street\_Address | Customer address | VCHAR2 | 50 | X(50) | N/A | Yes |  | Must be a valid address |
| City | Customer city | VCHAR2 | 50 | X(50) | N/A | Yes |  | Must be a valid city |
| State | Customer state | VCHAR2 | 2 | X(2) | N/A | Yes |  | Must be a valid state |
| Zip\_Code | Customer zip code | VCHAR2 | 10 | X(10) | N/A | Yes |  | Must be a valid zip code |
| Country | Customer country | VCHAR2 | 50 | X(50) | N/A | Yes |  | Must be a valid state |
| CustomerID | Primary key from Customer table | NUMBER | Sys Size | 9(-10^38 +1 to 10^38 –1) | 0 - 999… | Yes | FK | (Customer) Update - Cascades  Delete - Can’t be null |
| TypeID | Primary key from AddressType table | NUMBER | Sys Size | 9(-10^38 +1 to 10^38 –1) | 0 - 999… | Yes | FK | (AddressType) Update - Cascades  Delete - Can’t be null |

**SPECIAL RESTRICTIONS:** The Address table needs to be in the system as long as the Customer table exists, as it exists because of the Customer table. Address also needs to archive data for up to 20 years to keep up with government standards and maintain legality.

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

**NAME:** AddressType  
**DESCRIPTION:** This table contains data about the types of addresses in the database.

**RELATIONAL FORMAT:** Address( TypeID, Type\_Code )

**OTHER TABLES REFERENCED:** N/A

| Attribute Name | Contents | Type | Size | Format | Range | Req | Key | Restrictions |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| TypeID | Identity number for AddressType | NUMBER | Sys Size | 9(-10^38 +1 to 10^38 –1) | 0 - 999… | Yes | PK | Unique |
| Type\_Code | Type of address | VCHAR2 | 50 | X(50) | N/A | Yes |  |  |

**SPECIAL RESTRICTIONS:** The AddressType table needs to exist for the Address table, as the Address table depends on AddressType for its various types of addresses.

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

**NAME:** Region  
**DESCRIPTION:** This table contains data about the regions in the database.

**RELATIONAL FORMAT:** Region( RegionId, Region\_Name )

**OTHER TABLES REFERENCED:** N/A

| Attribute Name | Contents | Type | Size | Format | Range | Req | Key | Restrictions |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| RegionID | Identity number for AddressType | NUMBER | Sys Size | 9(-10^38 +1 to 10^38 –1) | 0 - 999… | Yes | PK | Unique |
| Region\_Name | Name of region | VCHAR2 | 50 | X(50) | N/A | Yes |  |  |

**SPECIAL RESTRICTIONS:** The Region table needs to exist for the Customer table, as the Customer table needs regions for its customer entries.

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

**NAME:** Customer

**DESCRIPTION:** This table contains data about customers in the sense of their personal data, such as names, emails, phone numbers, etc. Dates also need to be kept within these records as well because of promotional materials and deals that would be applied to customers, whether they’re long standing or new.

**RELATIONAL FORMAT:** Customer ( CustomerID, Customer\_Name, CN\_Name, Phone, Email, RegionID, Date\_Added, Date\_Updated )

**OTHER TABLES REFERENCED:** Address

| Attribute Name | Contents | Type | Size | Format | Range | Req | Key | Restrictions |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| CustomerID | Identity number for customers | NUMBER | Sys Size | 9(-10^38 +1 to 10^38 –1) | 0 - 999… | Yes | PK | Unique |
| Customer\_Name | Customer name | VCHAR2 | 100 | X(100) | N/A | Yes |  |  |
| CN\_Name | Contact name | VCHAR2 | 50 | X(50) | N/A | Yes |  |  |
| Phone | Customer phone number | VCHAR2 | 13 | X(13) | N/A | Yes |  | Must be a valid phone number |
| Email | Customer email address | VCHAR2 | 100 | X(100) | N/A | Yes |  | Must have a valid email provider |
| RegionID | Primary key from Region table | NUMBER | Sys Size | 9(-10^38 +1 to 10^38 –1) | 0 - 999… | Yes | FK | (Region) Update - Cascades  Delete - Can’t be null |
| Date\_Added | Date added into database | DATETIME | 9 | DD-MMM-YY | 01-JAN-00 - 31-DEC-99 | Yes |  | Must be a valid date within the range |
| Date\_Updated | Date updated in database (if not updated, remains same as date added) | DATETIME | 9 | DD-MMM-YY | 01-JAN-00 - 31-DEC-99 | No |  | Must be a valid date within the range and should be able to copy Date\_Added |

**SPECIAL RESTRICTIONS:** The Customer table needs to archive data as long as it is required by governmental entities for legal reasons.

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

**NAME:** Product\_Built

**DESCRIPTION:** This table holds a database of products offered by the company ArchBike. This maintains every quality of the products, such as their types, colors, sizes, etc. Various different entries may be needed for single models depending on whether or not they have different aspects like color, size, etc. The amount of each product is also listed within this table.

**RELATIONAL FORMAT:** Product ( ProductID, ModelID, ColorID, SizeID, OH\_Quantity, Initial\_Price )

**OTHER TABLES REFERENCED:** Product\_Size, Product\_Model, Product\_Size

| Attribute Name | Contents | Type | Size | Format | Range | Req | Key | Restrictions |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ProductID | Identity number for product | NUMBER | Sys Size | 9(-10^38 +1 to 10^38 –1) | 0 - 999… | Yes | PK | Unique |
| ModelID | Primary key from Product\_Model table | NUMBER | Sys Size | 9(-10^38 +1 to 10^38 –1) | 0 - 999… | Yes | FK | (Region) Update - Cascades  Delete - Can’t be null |
| ColorID | Primary key from Product\_Color table | NUMBER | Sys Size | 9(-10^38 +1 to 10^38 –1) | 0 - 999… | Yes | FK | (Region) Update - Cascades  Delete - Can’t be null |
| SizeID | Primary key from Product\_Size table | NUMBER | Sys Size | 9(-10^38 +1 to 10^38 –1) | 0 - 999… | Yes | FK | (Region) Update - Cascades  Delete - Can’t be null |
| OH\_Quantity | Total count of an individual product within the company | NUMBER | Sys Size | 9(-10^38 +1 to 10^38 –1) | 0 - 999… | Yes |  |  |
| Initial\_Price | Price of product before tax, discount, etc. | NUMBER | Sys Size | 9(-10^38 +1 to 10^38 –1) | 0 - 999… | Yes |  |  |

**SPECIAL RESTRICTIONS:** The product\_built table is going to be one of the most regularly altered tables, so it needs to be prepared in a way that it can constantly receive, edit, and archive multiple different entries for products, since there can be various prices for a single product.

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

**NAME:** Product\_Size

**DESCRIPTION:** This table is used to specifically track the sizes of products. This covers multiple copies of the same product and different products, as all bikes have various different sizes.

**RELATIONAL FORMAT:** Product\_Size ( SizeID, Product\_Size)  
**OTHER TABLES REFERENCED:** N/A

| Attribute Name | Contents | Type | Size | Format | Range | Req | Key | Restrictions |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| SizeID | Identity number for size | NUMBER | Sys Size | 9(-10^38 +1 to 10^38 –1) | 0 - 999… | Yes | PK | Unique |
| Product\_Size | The specific sizes of individual models | VCHAR2 | 20 | X(20) | N/A | Yes |  | Has to be a valid size bike |

**SPECIAL RESTRICTIONS:** The Product table is dependent on the Sizes table, so as long as Product exists, Sizes must also exist.

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

**NAME:** Product\_Color  
**DESCRIPTION:** This table is used to specifically track the colors of products. This covers multiple copies of the same product and different products, as some bikes have various different colors.

**RELATIONAL FORMAT:** Colors ( ColorID, Color\_Name )  
**OTHER TABLES REFERENCED:** N/A

| Attribute Name | Contents | Type | Size | Format | Range | Req | Key | Restrictions |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ColorID | Identity number for color | NUMBER | Sys Size | 9(-10^38 +1 to 10^38 –1) | 0 - 999… | Yes | PK | Unique |
| Color | The specific colors of individual models | VCHAR2 | 25 | X(25) | N/A | Yes |  |  |

**SPECIAL RESTRICTIONS:** This table needs to exist for Product to exist, because Product is dependent on it.

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

**NAME:** Category  
**DESCRIPTION:** This table is used to specifically track the categories of products. This covers multiple copies of the same product and different products, as some bike models have different categories.

**RELATIONAL FORMAT:** Categories ( CategoryID, Category\_Code, Category\_Name )

**OTHER TABLES REFERENCED:** N/A

| Attribute Name | Contents | Type | Size | Format | Range | Req | Key | Restrictions |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| CategoryID | Identity number for category | NUMBER | Sys Size | 9(-10^38 +1 to 10^38 –1) | 0 - 999… | Yes | PK | Unique |
| Category\_Code | The specific codes of categories | VCHAR2 | 4 | X(4) | N/A | Yes |  | Must be a valid category option |
| Category\_Name | Full title of each category | VCHAR2 | 50 | X(50) | N/A | Yes |  |  |

**SPECIAL RESTRICTIONS:** This table needs Product\_Model to exist, as the Product\_Model table is dependent on the Category table.

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

**NAME:** Product\_Model  
**DESCRIPTION:** This table is used to detail more about specific models of each product.

**RELATIONAL FORMAT:** Product\_Model ( ModelID, Model\_Code, CategoryID, Date\_Added, Date\_Stopped)

**OTHER TABLES REFERENCED:** Category

| Attribute Name | Contents | Type | Size | Format | Range | Req | Key | Restrictions |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ModelID | Identity number for model | NUMBER | Sys Size | 9(-10^38 +1 to 10^38 –1) | 0 - 999… | Yes | PK | Unique |
| Model\_Code | The specific codes of models | VCHAR2 | 10 | X(10) | N/A | Yes |  | Must be a valid category option |
| CategoryID | Primary key for category table | NUMBER | Sys Size | 9(-10^38 +1 to 10^38 –1) | 0 - 999… | Yes | FK | (Category) Update - Cascades  Delete - Can’t be null |
| Date\_Added | Date added in production | DATETIME | 9 | DD-MMM-YY | 01-JAN-00 - 31-DEC-99 | Yes |  | Must be a valid date within the range |
| Date\_Stopped | Date taken out of production | DATETIME | 9 | DD-MMM-YY | 01-JAN-00 - 31-DEC-99 | No |  | Must be a valid date within the range |

**SPECIAL RESTRICTIONS:** This table needs Category to exist and is needed by Product\_Built, as there is a chain of dependency there.

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

**NAME:** Model\_Text  
**DESCRIPTION:** This table contains different texts that relate to the product, such as a description of the product, start and stop dates of production, information about the product, etc.

**RELATIONAL FORMAT:** Model\_Text ( TextID, Description, Start\_Date, Stop\_Date, ModelID )  
**OTHER TABLES REFERENCED:** Product

| Attribute Name | Contents | Type | Size | Format | Range | Req | Key | Restrictions |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| TextID | Identity number for text | NUMBER | Sys Size | 9(-10^38 +1 to 10^38 –1) | 0 - 999… | Yes | PK | Unique |
| Description | Description for specific models | VCHAR2 | 50 | X(50) | N/A | Yes |  |  |
| Start\_Date | Start date of listing for the product | DATETIME | 9 | DD-MMM-YY | 01-JAN-00 - 31-DEC-99 | Yes |  | Must be a valid date within the range |
| Stop\_Date | Stop date of listing for the product | DATETIME | 9 | DD-MMM-YY | 01-JAN-00 - 31-DEC-99 | No |  | Must be a valid date within the range |
| ModelID | Primary key from Product table | NUMBER | Sys Size | 9(-10^38 +1 to 10^38 –1) | 0 - 999… | Yes | PK | (Product\_Model) Update - Cascades  Delete - Can’t be null |

**SPECIAL RESTRICTIONS:** This table is dependent on the Product\_Model table, so it can only exist as long as Product\_Model exists.

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

**NAME:** Inventory

**DESCRIPTION:** This table holds and maintains all products within the inventory of Arch Bike, the various models, the locations of them within warehouses that belong to Arch Bike, etc. This table is essentially more of an operational end table that works to help maintain structure to the physical inventory.

**RELATIONAL FORMAT:** Inventory ( INV\_ID, Aisle\_No, Row\_No, Serial\_No, Date\_Added, Status, WarehouseID, ProductID )  
**OTHER TABLES REFERENCED:** Product\_Built, Warehouse

| Attribute Name | Contents | Type | Size | Format | Range | Req | Key | Restrictions |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| InventoryID | Identity number for inventory | NUMBER | Sys Size | 9(-10^38 +1 to 10^38 –1) | 0 - 999… | Yes | PK | Unique |
| Aisle\_No | Aisle location for individual products | NUMBER | Sys Size | 9(-10^38 +1 to 10^38 –1) | 0 - 999… | Yes | PK | Must be a valid aisle location |
| Row\_No | Row location for individual products | NUMBER | Sys Size | 9(-10^38 +1 to 10^38 –1) | 0 - 999… | Yes | PK | Must be a valid row location |
| Serial\_No | Serial number identifier for individual products | NUMBER | Sys Size | 9(-10^38 +1 to 10^38 –1) | 0 - 999… | Yes | PK | Must be a valid serial number |
| Date\_Added | Date of listing adding for the product | DATETIME | 9 | DD-MMM-YY | 01-JAN-00 - 31-DEC-99 | Yes |  | Must be a valid date within the range |
| Status | Status of individual products within inventory | VCHAR2 | 500 | X(500) | N/A | Yes |  | Must be a valid status |
| WarehouseID | Primary key from Warehouse table | NUMBER | Sys Size | 9(-10^38 +1 to 10^38 –1) | 0 - 999… | Yes | PK | (Warehouse) Update - Cascades  Delete - Can’t be null |
| ProductID | Primary key from Product\_Built table | NUMBER | Sys Size | 9(-10^38 +1 to 10^38 –1) | 0 - 999… | Yes | PK | (Product\_Built) Update - Cascades  Delete - Can’t be null |

**SPECIAL RESTRICTIONS:** This table relates to Product\_Built and Warehouse, so it is dependent those tables and as long as ArchBike has products to sell, Inventory will need to exist so that these items can be stored.

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

**NAME:** Warehouse  
**DESCRIPTION:** This table holds information about individual warehouses such as their addresses, regions, warehouse numbers, etc. Each warehouse is used as a storage facility for the inventory of Arch Bike.  
**RELATIONAL FORMAT:** Warehouse ( WarehouseID, Warehouse\_Code, Street\_Address, City, State, Zip\_Code )  
**OTHER TABLES REFERENCED:** N/A

| Attribute Name | Contents | Type | Size | Format | Range | Req | Key | Restrictions |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| WarehouseID | Identity number for warehouse | NUMBER | Sys Size | 9(-10^38 +1 to 10^38 –1) | 0 - 999… | Yes | PK | Unique |
| Warehouse\_Code | Code listing for warehouse | VCHAR2 | 10 | X(10) | N/A | Yes |  |  |
| Street\_Address | Warehouse address | VCHAR2 | 50 | X(50) | N/A | Yes |  | Must be a valid address |
| City | Warehouse city | VCHAR2 | 50 | X(50) | N/A | Yes |  | Must be a valid city |
| State | Warehouse city | VCHAR2 | 2 | X(2) | N/A | Yes |  | Must be a valid state |
| Zip\_Code | Warehouse zip code | VCHAR2 | 10 | X(10) | N/A | Yes |  | Must be a valid zip code |

**SPECIAL RESTRICTIONS:** This table is required by Inventory to function, as WarehouseID is a foreign key used in Inventory

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

**NAME:** Price

**DESCRIPTION:** This table handles pricing for different products within the inventory of ArchBike. This is important as certain products can have multiple pricings based on their status, sales, etc.

**RELATIONAL FORMAT:** Price ( PriceID, Price, Start\_Date, End\_Date, Change\_Comm, InventoryID )  
**OTHER TABLES REFERENCED:** Inventory

| Attribute Name | Contents | Type | Size | Format | Range | Req | Key | Restrictions |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| PriceID | Identity number for price | NUMBER | Sys Size | 9(-10^38 +1 to 10^38 –1) | 0 - 999… | Yes | PK | Unique |
| Price | Prices listed for different models and products | NUMBER | Sys Size | 9(-10^38 +1 to 10^38 –1) | 0 - 999… | Yes |  |  |
| Start\_Date | Start date for each price of a product(s) | DATETIME | 9 | DD-MMM-YY | 01-JAN-00 - 31-DEC-99 | Yes |  | Must be a valid date within the range |
| End\_Date | End date for each price of a product(s) | DATETIME | 9 | DD-MMM-YY | 01-JAN-00 - 31-DEC-99 | No |  | Must be a valid date within the range |
| Change\_Comm | Comment for a price’s change reasoning | VCHAR2 | 250 | X(250) | N/A | No |  | Must contain valid text |
| InventoryID | Primary key from Inventory table | NUMBER | Sys Size | 9(-10^38 +1 to 10^38 –1) | 0 - 999… | Yes | FK | (Inventory) Update - Cascades  Delete - Can’t be null |

**SPECIAL RESTRICTION:** This table requires the inventory table to exist and function.

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

**NAME:** AccountReceivable  
**DESCRIPTION:** This table handles accounts receivable on the transaction end of the sales orders. It lists dates due for payment, gives a transaction ID, etc. This is important for billing customers.  
**RELATIONAL FORMAT:** Account\_Receivable ( Pay\_ID, TID, Date\_Recorded, Amount, Date\_Updated, Date\_Due, Balance\_Due )  
**OTHER TABLES REFERENCED:** Sales\_Transaction

| Attribute Name | Contents | Type | Size | Format | Range | Req | Key | Restrictions |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Pay\_ID | Identity number for accounts receivable | NUMBER | Sys Size | 9(-10^38 +1 to 10^38 –1) | 0 - 999… | Yes | PK | Unique |
| TID | Primary key from sales\_transaction table | NUMBER | Sys Size | 9(-10^38 +1 to 10^38 –1) | 0 - 999… | Yes | FK | (Sales\_Transaction) Update - Cascades  Delete - Can’t be null |
| Date\_Recorded | Date for whenever a payment is recorded | DATETIME | 9 | DD-MMM-YY | 01-JAN-00 - 31-DEC-99 | Yes |  | Must be a valid date |
| Amount | Total amount due | NUMBER | Sys Size | 9(-10^38 +1 to 10^38 –1) | 0 - 999… | Yes |  | Must be a valid amount |
| Date\_Updated | Date for when a payment is made or updated | DATETIME | 9 | DD-MMM-YY | 01-JAN-00 - 31-DEC-99 | No |  | Must be a valid date |
| Date\_Due | Date for when payment is due | DATETIME | 9 | DD-MMM-YY | 01-JAN-00 - 31-DEC-99 | Yes |  | Must be a valid date |
| Balance\_Due | Total amount left | NUMBER | Sys Size | 9(-10^38 +1 to 10^38 –1) | 0 - 999… | Yes |  | Must be a valid date |

**SPECIAL RESTRICTIONS:** This table is based on Sales\_Transaction, so it needs the primary key from Sales\_Transaction to exist and function properly. This table also could hold specific customer data so should reflect any important governmental restrictions needed for legality as well if necessary.

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

**NAME:** Sales\_Transaction

**DESCRIPTION:** This table handles the sales order interaction between the customer and the vendor. This lists various bits of information about the specific sale, such as the date it was ordered, the total amount, etc.

**RELATIONAL FORMAT:** Sales\_Transaction ( TID, TDate, TAmount, CustomerID )  
**OTHER TABLES REFERENCED:** Customer

| Attribute Name | Contents | Type | Size | Format | Range | Req | Key | Restrictions |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| TID | Identity number sales order | NUMBER | Sys Size | 9(-10^38 +1 to 10^38 –1) | 0 - 999… | Yes | PK | Unique |
| TDate | Date that the sales transaction was created | DATETIME | 9 | DD-MMM-YY | 01-JAN-00 - 31-DEC-99 | Yes |  | Must be a valid date within the range |
| TAmount | Total amount of the transaction | NUMBER | Sys Size | 9(-10^38 +1 to 10^38 –1) | 0 - 999… | Yes |  | Must be a valid date within the range |
| CustomerID | Primary key from Customer Table | NUMBER | Sys Size | 9(-10^38 +1 to 10^38 –1) | 0 - 999… | Yes | FK | (Customer) Update - Cascades  Delete - Can’t be null |

**SPECIAL RESTRICTIONS:** This table requires CustomerID from Customer, and is required by Transaction\_Line.

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

**NAME:** Transaction\_Line

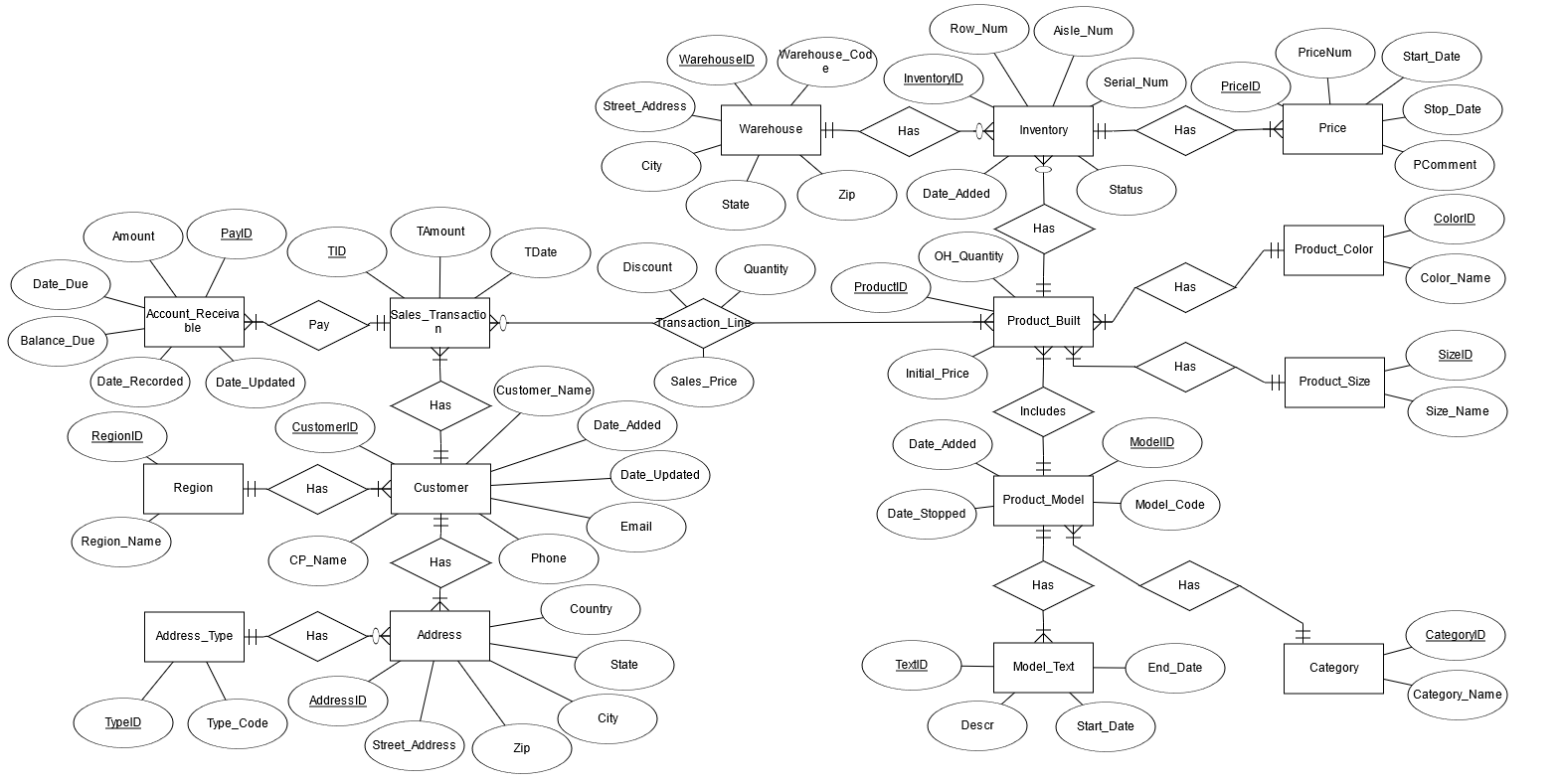
**DESCRIPTION:** This table handles pricing aspects of the sales order transaction such as sales price, unit price, total, etc. This is important for listing out the charges that a customer will be receiving for visual clarity.  
**RELATIONAL FORMAT:** Transaction\_Line ( ProductID, TID, Quantity, Sales\_Price, Discount)  
**OTHER TABLES REFERENCED:** Sales\_Transaction, Product\_Built

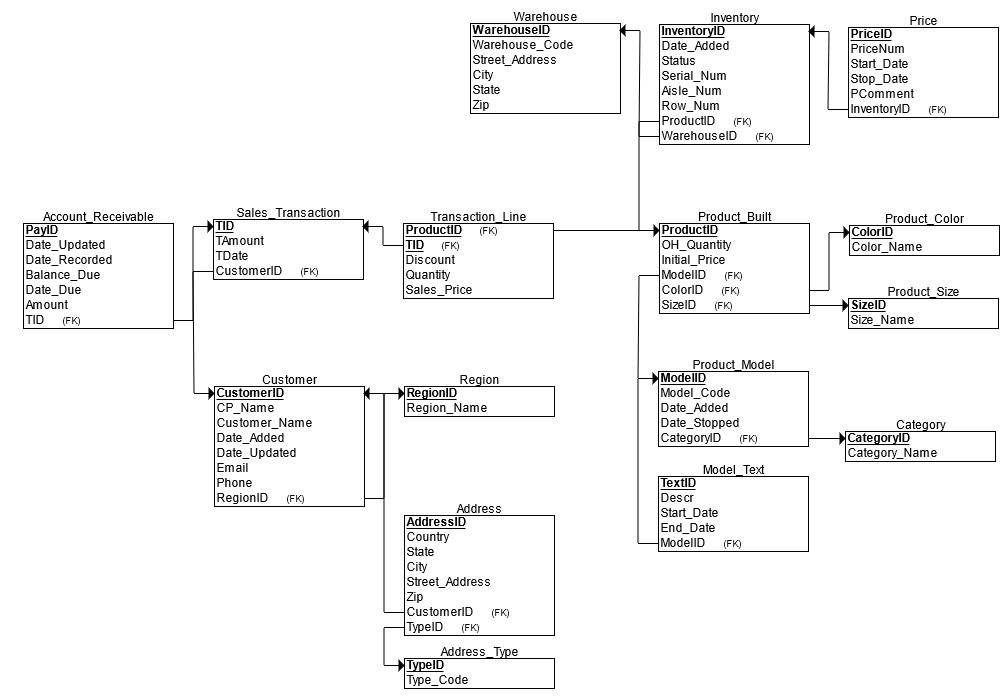
| Attribute Name | Contents | Type | Size | Format | Range | Req | Key | Restrictions |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ProductID | Primary key from Product table | NUMBER | Sys Size | 9(-10^38 +1 to 10^38 –1) | 0 - 999… | Yes | FK | (Product) Update - Cascades  Delete - Can’t be null |
| TID | Primary key from SalesOrder table | NUMBER | Sys Size | 9(-10^38 +1 to 10^38 –1) | 0 - 999… | Yes | FK | (SalesOrder) Update - Cascades  Delete - Can’t be null |
| Quantity | Quantity of a specific item in the transaction | NUMBER | Sys Size | 9(-10^38 +1 to 10^38 –1) | 0 - 999… | Yes |  | Must be a valid number |
| Sales\_Price | Sales price of a specific item in the transaction | NUMBER | Sys Size | 9(-10^38 +1 to 10^38 –1) | 0 - 999… | Yes |  | Must be a valid number |
| Discount | Discount applied to items in transaction | NUMBER | Sys Size | 9(-10^38 +1 to 10^38 –1) | 0 - 999… | Yes |  | Must be a valid number |

**SPECIAL RESTRICTIONS:** This table is dependent on Sales\_Transaciton and Product\_Built so those both need to exist for this one to exist. Because this deals with monetary information in relation to a customer, it should be stated that governmental regulations could add restrictions on archiving.

The following diagrams are an Entity Relationship Diagram as well as a Relational Schema that is used to show the structure of the tables in this database.

**Figure 16**

Entity Relational Diagram

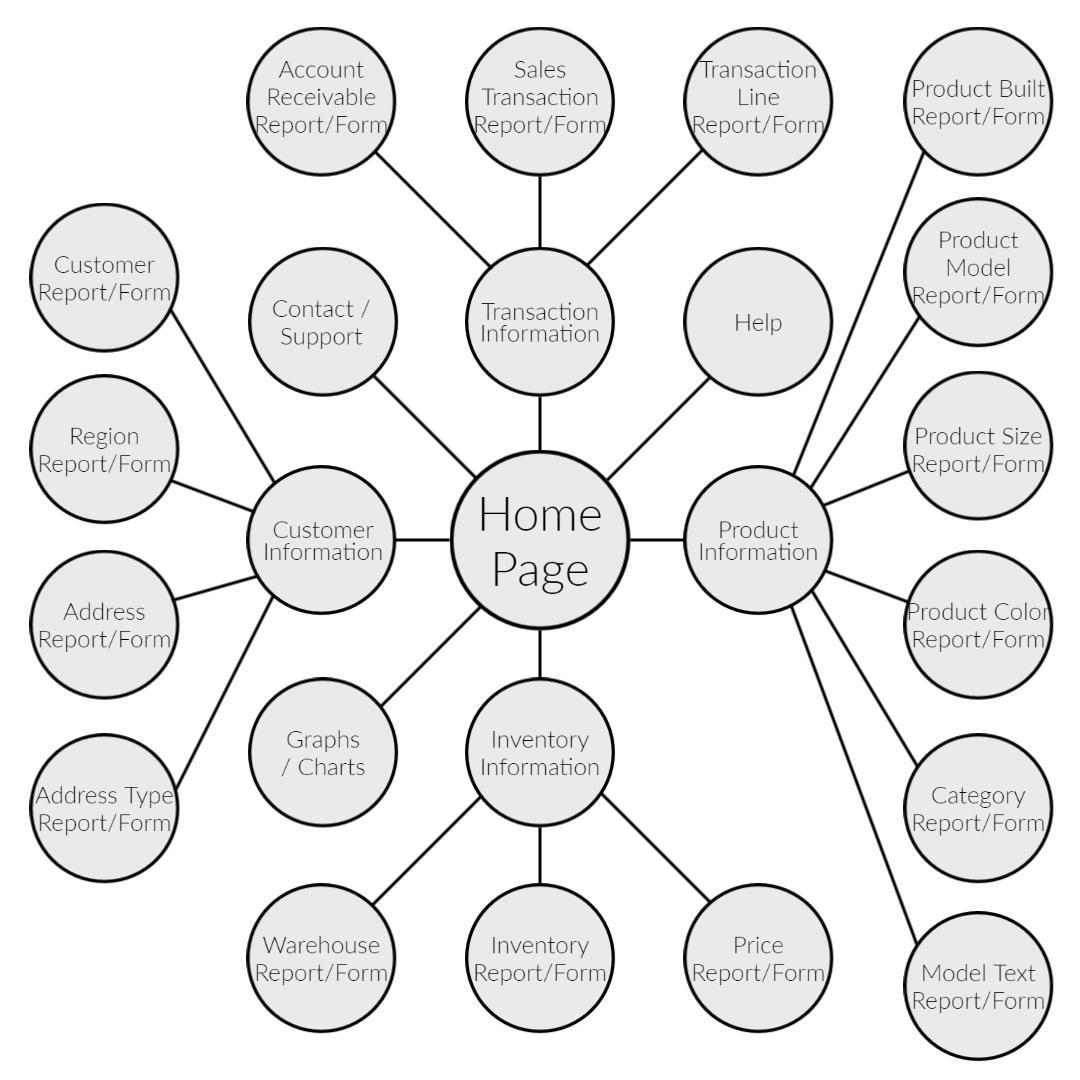
**Figure 17**  
Relationship Schema

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

Project Documentation:

This section is going to contain diagrams, code, and various other important features of the database from a technical perspective. These will not contain much text to pad it, but will be a simplistic look at the code and idea behind the structures of this database.

**Figure 17**  
Structural Diagram



(*Note.* This was created during a structural phase. It should be noted that there are 3 more nodes that stem from Graphs / Charts, those being Sales by Product, Sales by Category, and Sales by Customer)

**Table Creation SQL Code:**

-- Sales inventory Database CREATE.

-- 16 Tables

-- 15 Sequences

-- Triggers - for Auto-number and for create and/or update "RECORDS"

-- Date Created: 12/25/2022

-- Date Updated: 01/14/2023

--\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* CREATE TABLES, Sequence, and Triggers \*\*\*\*\*\*\*\*\*\*\*\*\*\*/

-- 1: REGION

CREATE TABLE REGION

(

RegionID NUMBER NOT NULL, -- Auto Number

Region\_Name VARCHAR2(50) NOT NULL,

CONSTRAINT REGION\_PK PRIMARY KEY (RegionID)

);

CREATE SEQUENCE REGION\_seq INCREMENT BY 1 START WITH 1 NOCACHE;

CREATE OR REPLACE TRIGGER INSERT\_REGION\_seq

BEFORE INSERT ON REGION

FOR EACH ROW

DECLARE

new\_id number;

BEGIN

SELECT REGION\_SEQ.nextval INTO new\_id FROM dual;

:new.REGIONID := new\_id;

END;

/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

-- 2: Customer

CREATE TABLE CUSTOMER

(

CustomerID NUMBER NOT NULL,

Customer\_Name VARCHAR2(100) NOT NULL,

CP\_Name VARCHAR2(50) NOT NULL,

Phone VARCHAR2(13) NOT NULL,

Email VARCHAR2(100) NOT NULL,

RegionID NUMBER NOT NULL,

Date\_Added DATE,

Date\_Updated DATE,

CONSTRAINT CUSTOMER\_PK PRIMARY KEY (CustomerID),

CONSTRAINT REGION\_CUSTOMER\_FK FOREIGN KEY (REGIONID) REFERENCES REGION(REGIONID)

);

CREATE SEQUENCE CUSTOMER\_SEQ INCREMENT BY 1 START WITH 1 NOCACHE;

CREATE OR REPLACE TRIGGER INSERT\_CUSTOMER\_SEQ

BEFORE INSERT ON CUSTOMER

FOR EACH ROW

DECLARE

new\_id number;

BEGIN

SELECT CUSTOMER\_SEQ.nextval INTO new\_id FROM dual;

:new.CustomerID := new\_id;

END;

/

CREATE OR REPLACE TRIGGER INSERT\_CUSTOMER\_Date

BEFORE INSERT ON CUSTOMER

FOR EACH ROW

BEGIN

:NEW.Date\_Added := SYSDATE;

END;

/

CREATE OR REPLACE TRIGGER UPDATE\_CUSTOMER\_Date

BEFORE UPDATE ON CUSTOMER

FOR EACH ROW

BEGIN

:NEW.Date\_Updated := SYSDATE;

END;

/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

-- 3: Category

CREATE TABLE CATEGORY

(

CategoryID Number NOT NULL,

Category\_Code VARCHAR2(4),

Category\_Name VARCHAR2(50) NOT NULL,

CONSTRAINT CATEGORY\_PK PRIMARY KEY (CategoryID)

);

CREATE SEQUENCE CATEGORY\_seq INCREMENT BY 1 START WITH 1 NOCACHE;

CREATE OR REPLACE TRIGGER INSERT\_CATEGORY\_SEQ

BEFORE INSERT ON CATEGORY

FOR EACH ROW

DECLARE

new\_id number;

BEGIN

SELECT CATEGORY\_seq.nextval INTO new\_id FROM dual;

:new.CATEGORYID := new\_id;

END;

/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

-- 4: WAREHOUSE

CREATE TABLE Warehouse

(

Warehouseid NUMBER NOT NULL,

Warehouse\_code varchar2(10),

Street\_Address VARCHAR2(50) NOT NULL,

City VARCHAR2(50) NOT NULL,

State VARCHAR2(2) NOT NULL,

ZIP\_Code VARCHAR2(10) NOT NULL,

CONSTRAINT WAREHOUSE\_PK PRIMARY KEY (Warehouseid)

);

CREATE SEQUENCE Warehouse\_seq INCREMENT BY 1 START WITH 1 NOCACHE;

CREATE OR REPLACE TRIGGER INSERT\_Warehouse\_SEQ

BEFORE INSERT ON Warehouse

FOR EACH ROW

DECLARE

new\_id number;

BEGIN

SELECT Warehouse\_SEQ.nextval INTO new\_id FROM dual;

:new.WarehouseID := new\_id;

END;

/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

-- 5: PRODUCT\_MODEL

CREATE TABLE Product\_Model

(

ModelID NUMBER NOT NULL,

Model\_Code VARCHAR2(10) NOT NULL,

CategoryID NUMBER NOT NULL,

Date\_\_Added DATE NULL,

Date\_Stopped DATE NULL,

CONSTRAINT PRODUCT\_MODEL\_PK PRIMARY KEY (ModelID),

CONSTRAINT PRODUCT\_MODEL\_CATEGORY\_FK FOREIGN KEY (CategoryID) REFERENCES Category(CategoryID)

);

CREATE SEQUENCE PRODUCT\_MODEL\_seq INCREMENT BY 1 START WITH 1 NOCACHE;

CREATE OR REPLACE TRIGGER INSERT\_PRODUCT\_MODEL\_seq

BEFORE INSERT ON PRODUCT\_MODEL

FOR EACH ROW

DECLARE

new\_id number;

BEGIN

SELECT PRODUCT\_MODEL\_SEQ.nextval INTO new\_id FROM dual;

:new.MODELID := new\_id;

END;

/

CREATE OR REPLACE TRIGGER INSERT\_PRODUCT\_MODEL\_Date

BEFORE INSERT ON PRODUCT\_MODEL

FOR EACH ROW

BEGIN

:NEW.Date\_\_Added:= SYSDATE;

END;

/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

--6: MODEL\_TEXT

CREATE TABLE Model\_Text

(

TextID NUMBER NOT NULL,

Description VARCHAR2(50) NOT NULL,

ModelID NUMBER NOT NULL,

Start\_Date DATE,

End\_Date DATE,

CONSTRAINT MODEL\_TEXT\_PK PRIMARY KEY (TextID),

CONSTRAINT MODEL\_TEXT\_PRODUCT\_MODEL\_FK FOREIGN KEY (ModelID) REFERENCES Product\_Model(ModelID)

);

CREATE SEQUENCE MODEL\_TEXT\_seq INCREMENT BY 1 START WITH 1 NOCACHE;

CREATE OR REPLACE TRIGGER INSERT\_MODEL\_TEXT\_seq

BEFORE INSERT ON MODEL\_TEXT

FOR EACH ROW

DECLARE

new\_id number;

BEGIN

SELECT MODEL\_TEXT\_SEQ.nextval INTO new\_id FROM dual;

:new.TEXTID := new\_id;

END;

/

CREATE OR REPLACE TRIGGER INSERT\_MODEL\_TEXT\_Date

BEFORE INSERT ON MODEL\_TEXT

FOR EACH ROW

BEGIN

:NEW.START\_DATE:= SYSDATE;

END;

/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

--7: PRODUCT\_COLOR

CREATE TABLE Product\_Color

(

ColorID NUMBER NOT NULL,

Color\_Name VARCHAR2(50) NOT NULL,

CONSTRAINT PRODUCT\_COLOR\_PK PRIMARY KEY (ColorID)

);

CREATE SEQUENCE PRODUCT\_COLOR\_seq INCREMENT BY 1 START WITH 1 NOCACHE;

CREATE OR REPLACE TRIGGER INSERT\_PRODUCT\_COLOR\_seq

BEFORE INSERT ON PRODUCT\_COLOR

FOR EACH ROW

DECLARE

new\_id number;

BEGIN

SELECT PRODUCT\_COLOR\_SEQ.nextval INTO new\_id FROM dual;

:new.COLORID := new\_id;

END;

/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

--8 PRODUCT\_SIZE

CREATE TABLE Product\_Size

(

SizeID NUMBER NOT NULL,

PRODUCT\_Size VARCHAR2(20) NULL,

CONSTRAINT PRODUCT\_SIZE\_PK PRIMARY KEY (SizeID)

);

CREATE SEQUENCE PRODUCT\_SIZE\_seq INCREMENT BY 1 START WITH 1 NOCACHE;

CREATE OR REPLACE TRIGGER INSERT\_PRODUCT\_SIZE\_seq

BEFORE INSERT ON PRODUCT\_SIZE

FOR EACH ROW

DECLARE

new\_id number;

BEGIN

SELECT PRODUCT\_SIZE\_SEQ.nextval INTO new\_id FROM dual;

:new.SIZEID:= new\_id;

END;

/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

--9: ADDRESS\_TYPE

CREATE TABLE Address\_Type

(

TypeID NUMBER NOT NULL,

Type\_Code VARCHAR2(50) NOT NULL,

CONSTRAINT ADDRESS\_TYPE\_PK PRIMARY KEY (TypeID)

);

CREATE SEQUENCE ADDRESS\_TYPE\_seq INCREMENT BY 1 START WITH 1 NOCACHE;

CREATE OR REPLACE TRIGGER INSERT\_ADDRESS\_TYPE\_seq

BEFORE INSERT ON ADDRESS\_TYPE

FOR EACH ROW

DECLARE

new\_id number;

BEGIN

SELECT ADDRESS\_Type\_SEQ.nextval INTO new\_id FROM dual;

:new.TYPEID := new\_id;

END;

/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

--10: ADDRESS

CREATE TABLE Address

(

AddressID NUMBER NOT NULL, -- Auto Number

Street\_Address VARCHAR2(50) NOT NULL,

City VARCHAR2(50) NOT NULL,

State VARCHAR2(2) NOT NULL,

ZIP\_Code VARCHAR2(10) NOT NULL,

Country VARCHAR2 (50) NOT NULL,

CustomerID NUMBER NOT NULL,

TypeID NUMBER NOT NULL,

CONSTRAINT Address\_PK PRIMARY KEY (AddressID),

CONSTRAINT ADDRESS\_CUSTOMER\_FK FOREIGN KEY (CustomerID) REFERENCES CUSTOMER(CustomerID),

CONSTRAINT ADDRESS\_TYPE\_ADDRESS\_FK FOREIGN KEY(TYPEID) REFERENCES ADDRESS\_TYPE(TYPEID)

);

CREATE SEQUENCE Address\_seq INCREMENT BY 1 START WITH 1 NOCACHE;

CREATE OR REPLACE TRIGGER INSERT\_ADDRESS\_SEQ

BEFORE INSERT ON ADDRESS

FOR EACH ROW

DECLARE

new\_id number;

BEGIN

SELECT ADDRESS\_SEQ.nextval INTO new\_id FROM dual;

:new.ADDRESSID := new\_id;

END;

/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

--11 SALES\_TRANSACTION

CREATE TABLE SALES\_TRANSACTION

(

TID NUMBER NOT NULL, -- Auto Number

TDate Date NOT NULL,

TAmount Number (17,2) NULL,

CustomerID NUMBER NOT NULL,

CONSTRAINT SALES\_TRANSACTION\_PK PRIMARY KEY (TID),

CONSTRAINT SALES\_TRANSACTION\_CUSTOMER\_FK FOREIGN KEY(CustomerID) REFERENCES CUSTOMER(CustomerID)

);

CREATE SEQUENCE SALES\_TRANSACTION\_seq INCREMENT BY 1 START WITH 1 NOCACHE;

CREATE OR REPLACE TRIGGER INSERT\_SALES\_TRANSACTION\_seq

BEFORE INSERT ON SALES\_TRANSACTION

FOR EACH ROW

DECLARE

new\_id number;

BEGIN

SELECT SALES\_TRANSACTION\_SEQ.nextval INTO new\_id FROM dual;

:new.TID := new\_id;

END;

/

CREATE OR REPLACE TRIGGER INSERT\_Transaction\_Date

BEFORE INSERT ON SALES\_TRANSACTION

FOR EACH ROW

BEGIN

:NEW.TDate := SYSDATE;

END;

/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

--12: -- 9: ACCOUNT\_RECEIVABLE

CREATE TABLE Account\_Receivable

(

PayID NUMBER NOT NULL, -- Auto Number

TID NUMBER NOT NULL,

Date\_Recorded DATE NOT NULL,

Amount NUMBER(17,2) NOT NULL,

Date\_updated DATE NULL,

Date\_Due DATE NOT NULL,

Balance\_Due NUMBER(9,2) NOT NULL,

CONSTRAINT Account\_Receivable\_PK PRIMARY KEY(PayID),

CONSTRAINT Account\_Receivable\_SALES\_TRANSACTION\_FK FOREIGN KEY(TID) REFERENCES SALES\_TRANSACTION(TID)

);

CREATE SEQUENCE Account\_Receivable\_seq INCREMENT BY 1 START WITH 1 NOCACHE;

CREATE OR REPLACE TRIGGER INSERT\_Account\_Receivable\_seq

BEFORE INSERT ON Account\_Receivable

FOR EACH ROW

DECLARE

new\_id number;

BEGIN

SELECT ACCOUNT\_Receivable\_SEQ.nextval INTO new\_id FROM dual;

:new.PAYID := new\_id;

END;

/

CREATE OR REPLACE TRIGGER INSERT\_ACCOUNT\_Receivable\_Date

BEFORE INSERT ON ACCOUNT\_Receivable

FOR EACH ROW

BEGIN

:NEW.Date\_Recorded := SYSDATE;

END;

/

CREATE OR REPLACE TRIGGER UPDATE\_ACCOUNT\_Receivable\_Date

BEFORE UPDATE ON ACCOUNT\_Receivable

FOR EACH ROW

BEGIN

:NEW.Date\_updated := SYSDATE;

END;

/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

--13: PRODUCT\_BUILT

CREATE TABLE Product\_Built

(

ProductID NUMBER NOT NULL,

ModelID NUMBER NOT NULL,

ColorID NUMBER NULL,

SizeID NUMBER NULL,

OH\_Quantity NUMBER NOT NULL,

Initial\_Price NUMBER (9,2) NOT NULL,

CONSTRAINT PRODUCT\_BUILT PRIMARY KEY (ProductID),

CONSTRAINT PRODUCT\_BUILT\_COLOR\_FK FOREIGN KEY (ColorID) REFERENCES Product\_Color(ColorID),

CONSTRAINT PRODUCT\_BUILT\_SIZE\_FK FOREIGN KEY (SizeID) REFERENCES Product\_Size(SizeID),

CONSTRAINT PRODUCT\_BUILT\_MODEL\_FK FOREIGN KEY (ModelID) REFERENCES Product\_Model(ModelID)

);

CREATE SEQUENCE PRODUCT\_BUILT\_seq INCREMENT BY 1 START WITH 1 NOCACHE;

CREATE OR REPLACE TRIGGER INSERT\_PRODUCT\_BUILT\_seq

BEFORE INSERT ON PRODUCT\_BUILT

FOR EACH ROW

DECLARE

new\_id number;

BEGIN

SELECT PRODUCT\_BUILT\_SEQ.nextval INTO new\_id FROM dual;

:new.ProductID := new\_id;

END;

/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

--14: TRANSACTION\_LINE

CREATE TABLE Transaction\_Line

(

TID NUMBER NOT NULL,

ProductID NUMBER NOT NULL,

Quantity NUMBER NOT NULL,

Sales\_Price NUMBER(9,2) NOT NULL,

Discount NUMBER(7,2) NOT NULL,

CONSTRAINT TRANSACTION\_LINE\_PK PRIMARY KEY (TID, ProductID),

CONSTRAINT TRANSACTION\_LINE\_SLAES\_FK FOREIGN KEY (TID) REFERENCES Sales\_Transaction(TID),

CONSTRAINT TRANSACTION\_LINE\_PRODUCT\_FK FOREIGN KEY (ProductID) REFERENCES Product\_Built(ProductID)

);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

--15 INVENTORY

CREATE TABLE Inventory

(

InventoryID NUMBER NOT NULL,

Serial\_No NUMBER NOT NULL,

Aisle\_No NUMBER NOT NULL,

Row\_No NUMBER NOT NULL,

Date\_Added DATE NOT NULL,

Status VARCHAR2(500) NULL,

WarehouseID NUMBER NOT NULL,

ProductID NUMBER NULL,

CONSTRAINT INVENTORY\_PK PRIMARY KEY (InventoryID),

CONSTRAINT INVENTORY\_WAREHOUSE\_FK FOREIGN KEY (WarehouseID) REFERENCES Warehouse(WarehouseID),

CONSTRAINT INVENTORY\_PRODUCT\_BUILT\_FK FOREIGN KEY (ProductID) REFERENCES Product\_Built(ProductID)

);

CREATE SEQUENCE INVENTORY\_seq INCREMENT BY 1 START WITH 1 NOCACHE;

CREATE OR REPLACE TRIGGER INSERT\_Inventory\_seq

BEFORE INSERT ON Inventory

FOR EACH ROW

DECLARE

new\_id number;

BEGIN

SELECT INVENTORY\_SEQ.nextval INTO new\_id FROM dual;

:new.InventoryID := new\_id;

END;

/

CREATE OR REPLACE TRIGGER INSERT\_INVENTORY\_Date

BEFORE INSERT ON INVENTORY

FOR EACH ROW

BEGIN

:NEW.Date\_Added := SYSDATE;

END;

/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

--16: PRICE

CREATE TABLE Price

(

Priceid NUMBER NOT NULL,

Price NUMBER(9,2) NOT NULL,

Start\_Date DATE NOT NULL,

Stop\_Date DATE,

Change\_Comm VARCHAR2(250),

InventoryID NUMBER NOT NULL,

CONSTRAINT PRICE\_PK PRIMARY KEY (Priceid),

CONSTRAINT PRICE\_INVENTORY\_FK FOREIGN KEY (InventoryID) REFERENCES Inventory(InventoryID)

);

CREATE SEQUENCE PRICE\_seq INCREMENT BY 1 START WITH 1 NOCACHE;

CREATE OR REPLACE TRIGGER INSERT\_PRICE\_seq

BEFORE INSERT ON PRICE

FOR EACH ROW

DECLARE

new\_id number;

BEGIN

SELECT PRICE\_SEQ.nextval INTO new\_id FROM dual;

:new.PRICEID := new\_id;

END;

/

CREATE OR REPLACE TRIGGER INSERT\_PRICE\_Date

BEFORE INSERT ON PRICE

FOR EACH ROW

BEGIN

:NEW.Start\_Date := SYSDATE;

END;

/

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

**Table Data Population SQL Code:**

-- Sales inventory Database Load

-- 16 Tables

-- 15 Sequences

-- Date Created: 12/26/2022

-- Date Updated: 01/14/2023

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Insert into Region (Region\_Name) values('MidWest');

Insert into Region (Region\_Name) values('NorthWest');

Insert into Region (Region\_Name) values('South');

Insert into Region (Region\_Name) values('SouthWest');

Insert into Region (Region\_Name) values('SouthEast');

Insert into Region (Region\_Name) values('New England');

Insert into Region (Region\_Name) values('Great Lake');

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Insert into CUSTOMER (Customer\_Name,CP\_Name,Phone,Email,RegionID)

values('Airport Bikes', 'Tina', '622-091-9900','tina234@gmail.com', 1);

Insert into CUSTOMER (Customer\_Name,CP\_Name,Phone,Email,RegionID)

values('Capital Bikes','Tony','601-234-0911', 'tony2001@att.com', 4);

Insert into CUSTOMER (Customer\_Name,CP\_Name,Phone,Email,RegionID)

values('Alster Cycling','Pam','411-001-2145', 'moblig@icloud.com', 3);

Insert into CUSTOMER (Customer\_Name,CP\_Name,Phone,Email,RegionID)

values('Run Swim Bike','Elly','314-628-0032', 'macsmenace@ruswbi.net', 1);

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Insert Into Address\_Type (Type\_Code) values('Home');

Insert Into Address\_Type (Type\_Code) values('Billing');

Insert Into Address\_Type (Type\_Code) values('Shipping');

Insert Into Address\_Type (Type\_Code) values('Home/Billing');

Insert Into Address\_Type (Type\_Code) values('Home/Shipping');

Insert Into Address\_Type (Type\_Code) values('Billing/Shipping');

Insert Into Address\_Type (Type\_Code) values('Home/Billing/Shipping');

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Insert Into Address (Street\_Address, City, State, ZIP\_Code, Country, CustomerID, TypeID)

Values('45020 Aviation Drive','Saint Louis','MO','63166','US',1,1);

Insert Into Address (Street\_Address, City, State, ZIP\_Code, Country, CustomerID, TypeID)

values('719 41 North Street','Saint Louis','MO','63217','US',1, 2);

Insert Into Address (Street\_Address, City, State, ZIP\_Code, Country, CustomerID, TypeID)

values('5698 Elmonte Drive','Saint Louis','MO','63132','US',1,3);

Insert Into Address (Street\_Address, City, State, ZIP\_Code, Country, CustomerID, TypeID)

values('3235 Mi Casa Court','Birmingham','AL','35203','US',2,7);

Insert Into Address (Street\_Address, City, State, ZIP\_Code, Country, CustomerID, TypeID)

values('5698 Sunshine Drive','Chicago', 'IL','60610','US',3,1);

Insert Into Address (Street\_Address, City, State, ZIP\_Code, Country, CustomerID, TypeID)

values('12 State Street','Chicago','IL','61610','US', 3, 6);

Insert Into Address (Street\_Address, City, State, ZIP\_Code, Country, CustomerID, TypeID)

values('2530 Willow Court','Clearwater','FL','33755','US',4,1);

Insert Into Address (Street\_Address, City, State, ZIP\_Code, Country, CustomerID, TypeID)

values('2530 Willow Court','Clearwater','FL','33755','US',4,2);

Insert Into Address (Street\_Address, City, State, ZIP\_Code, Country, CustomerID, TypeID)

values('2530 Willow Court','Clearwater','FL','33755','US',4,3);

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Insert Into CATEGORY(Category\_Code,Category\_Name) values('ACC','Accessories');

Insert Into CATEGORY(Category\_Code,Category\_Name) values('TOU','Touring Bikes');

Insert Into CATEGORY(Category\_Code,Category\_Name) values('ROB','Road Bikes');

Insert Into CATEGORY(Category\_Code,Category\_Name) values('ORB','OffRoad Bikes');

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Insert Into Product\_Model(Model\_Code,CategoryID) Values('PRTR1000',2);

Insert Into Product\_Model(Model\_Code,CategoryID) Values('DXTR1000',2);

Insert Into Product\_Model(Model\_Code,CategoryID) Values('DXRD2000',3);

Insert Into Product\_Model(Model\_Code,CategoryID) Values('ORHT2000',4);

Insert Into Product\_Model(Model\_Code,CategoryID) Values('SHRT1000',1);

Insert Into Product\_Model(Model\_Code,CategoryID) Values('RHMT1000',1);

Insert Into Product\_Model(Model\_Code,CategoryID) Values('FAID1000',1);

Insert Into Product\_Model(Model\_Code,CategoryID) Values('PRRD1000',3);

Insert Into Product\_Model(Model\_Code,CategoryID) Values('PUMP1000',1);

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Insert Into Model\_Text(Description, ModelID) Values('Professional Touring Bike',1);

Insert Into Model\_Text(Description, ModelID) Values('Deluxe Touring Bike',2);

Insert Into Model\_Text(Description, ModelID) Values('Road Bike Alu SRAM',3);

Insert Into Model\_Text(Description, ModelID) Values('Off Road Bike Hard Tail SRAM',4);

Insert Into Model\_Text(Description, ModelID) Values('T-shirt',5);

Insert Into Model\_Text(Description, ModelID) Values('Road Helmet',6);

Insert Into Model\_Text(Description, ModelID) Values('First Aid Kit',7);

Insert Into Model\_Text(Description, ModelID) Values('Road Bike Carbon Shimano',8);

Insert Into Model\_Text(Description, ModelID) Values('Air Pump',9);

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Insert Into Warehouse(warehouse\_code, Street\_Address,City,State,ZIP\_Code) values('ASTL','13695 Rider Trail N','Earth City','MO','63045');

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Insert into Product\_Color(Color\_Name) values('Silver');

Insert into Product\_Color(Color\_Name) values('Blue');

Insert into Product\_Color(Color\_Name) values('Black');

Insert into Product\_Color(Color\_Name) values('Red');

Insert into Product\_Color(Color\_Name) values('No Color');

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Insert into Product\_Size (PRODUCT\_Size) values('28');

Insert into Product\_Size (PRODUCT\_Size) values('30');

Insert into Product\_Size (PRODUCT\_Size) values('32');

Insert into Product\_Size (PRODUCT\_Size) values('No Size');

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Insert Into Product\_Built(ModelID,ColorID,SizeID,OH\_Quantity, Initial\_Price) values(1,1,1,10,2300);

Insert Into Product\_Built(ModelID,ColorID,SizeID,OH\_Quantity, Initial\_Price) values(1,1,2,10,2250);

Insert Into Product\_Built(ModelID,ColorID,SizeID,OH\_Quantity, Initial\_Price) values(1,2,1,10,2350);

Insert Into Product\_Built(ModelID,ColorID,SizeID,OH\_Quantity, Initial\_Price) values(1,2,2,5,2350);

Insert Into Product\_Built(ModelID,ColorID,SizeID,OH\_Quantity, Initial\_Price) values(1,3,3,5,2310);

Insert Into Product\_Built(ModelID,ColorID,SizeID,OH\_Quantity, Initial\_Price) values(2,1,1,10,2000);

Insert Into Product\_Built(ModelID,ColorID,SizeID,OH\_Quantity, Initial\_Price) values(2,1,2,5, 2030);

Insert Into Product\_Built(ModelID,ColorID,SizeID,OH\_Quantity, Initial\_Price) values(2,1,3,10,2035);

Insert Into Product\_Built(ModelID,ColorID,SizeID,OH\_Quantity, Initial\_Price) values(2,2,1,5, 1950);

Insert Into Product\_Built(ModelID,ColorID,SizeID,OH\_Quantity, Initial\_Price) values(2,3,2,10, 1850);

Insert Into Product\_Built(ModelID,ColorID,SizeID,OH\_Quantity, Initial\_Price) values(3,3,2,20, 3000);

Insert Into Product\_Built(ModelID,ColorID,SizeID,OH\_Quantity, Initial\_Price) values(3,3,3,5, 3300);

Insert Into Product\_Built(ModelID,ColorID,SizeID,OH\_Quantity, Initial\_Price) values(3,2,1,5, 3350);

Insert Into Product\_Built(ModelID,ColorID,SizeID,OH\_Quantity, Initial\_Price) values(4,1,1,10,3900);

Insert Into Product\_Built(ModelID,ColorID,SizeID,OH\_Quantity, Initial\_Price) values(4,1,2,10, 3950);

Insert Into Product\_Built(ModelID,ColorID,SizeID,OH\_Quantity, Initial\_Price) values(4,3,1,5, 3850);

Insert Into Product\_Built(ModelID,ColorID,SizeID,OH\_Quantity, Initial\_Price) values(4,3,2,5, 3900);

Insert Into Product\_Built(ModelID,ColorID,SizeID,OH\_Quantity, Initial\_Price) values(5,5,4,15, 80);

Insert Into Product\_Built(ModelID,ColorID,SizeID,OH\_Quantity, Initial\_Price) values(6,5,4,15, 120);

Insert Into Product\_Built(ModelID,ColorID,SizeID,OH\_Quantity, Initial\_Price) values(7,5,4,10, 55);

Insert Into Product\_Built(ModelID,ColorID,SizeID,OH\_Quantity, Initial\_Price) values(8,1,1,5, 4500);

Insert Into Product\_Built(ModelID,ColorID,SizeID,OH\_Quantity, Initial\_Price) values(8,1,2,10, 4000);

Insert Into Product\_Built(ModelID,ColorID,SizeID,OH\_Quantity, Initial\_Price) values(8,1,3,5, 4700);

Insert Into Product\_Built(ModelID,ColorID,SizeID,OH\_Quantity, Initial\_Price) values(8,2,1,5, 4700);

Insert Into Product\_Built(ModelID,ColorID,SizeID,OH\_Quantity, Initial\_Price) values(8,2,2,5, 5000);

Insert Into Product\_Built(ModelID,ColorID,SizeID,OH\_Quantity, Initial\_Price) values(9,5,4,10, 20);

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- INVENTORY

DECLARE

i number(3);

SN number ;

BEGIN

-- productID 1

SN := 1234560;

<< loop >>

FOR i IN 1..10 LOOP

SN := SN + 1;

Insert into Inventory (Serial\_No, Aisle\_No, Row\_No, WarehouseID, ProductID)

values(SN,1,1,1,1);

END loop;

-- productID 2

<< loop >>

FOR i IN 1..10 LOOP

SN := SN + 1;

Insert into Inventory (Serial\_No, Aisle\_No, Row\_No, WarehouseID, ProductID)

values(SN,1,2,1,2);

END loop;

-- productID 3

<< loop >>

FOR i IN 1..10 LOOP

SN := SN + 1;

Insert into Inventory (Serial\_No, Aisle\_No, Row\_No, WarehouseID, ProductID)

values(SN,1,3,1,3);

END loop;

-- productID 4

<< loop >>

FOR i IN 1..5 LOOP

SN := SN + 1;

Insert into Inventory (Serial\_No, Aisle\_No, Row\_No, WarehouseID, ProductID)

values(SN,1,4,1,4);

END loop;

-- productID 5

<< loop >>

FOR i IN 1..5 LOOP

SN := SN + 1;

Insert into Inventory (Serial\_No, Aisle\_No, Row\_No, WarehouseID, ProductID)

values(SN,1,5,1,5);

END loop;

-- productID 6

<< loop >>

FOR i IN 1..10 LOOP

SN := SN + 1;

Insert into Inventory (Serial\_No, Aisle\_No, Row\_No, WarehouseID, ProductID)

values(SN,1,6,1,6);

END loop;

-- productID 7

<< loop >>

FOR i IN 1..5 LOOP

SN := SN + 1;

Insert into Inventory (Serial\_No, Aisle\_No, Row\_No, WarehouseID, ProductID)

values(SN,1,7,1,7);

END loop;

-- productID 8

<< loop >>

FOR i IN 1..10 LOOP

SN := SN + 1;

Insert into Inventory (Serial\_No, Aisle\_No, Row\_No, WarehouseID, ProductID)

values(SN,1,8,1,8);

END loop;

-- productID 9

<< loop >>

FOR i IN 1..5 LOOP

SN := SN + 1;

Insert into Inventory (Serial\_No, Aisle\_No, Row\_No, WarehouseID, ProductID)

values(SN,1,9,1,9);

END loop;

-- productID 10

<< loop >>

FOR i IN 1..10 LOOP

SN := SN + 1;

Insert into Inventory (Serial\_No, Aisle\_No, Row\_No, WarehouseID, ProductID)

values(SN,1,10,1,10);

END loop;

-- productID 11

<< loop >>

FOR i IN 1..20 LOOP

SN := SN + 1;

Insert into Inventory (Serial\_No, Aisle\_No, Row\_No, WarehouseID, ProductID)

values(SN,2,1,1,11);

END loop;

-- productID 12

<< loop >>

FOR i IN 1..5 LOOP

SN := SN + 1;

Insert into Inventory (Serial\_No, Aisle\_No, Row\_No, WarehouseID, ProductID)

values(SN,2,2,1,12);

END loop;

-- productID 13

<< loop >>

FOR i IN 1..5 LOOP

SN := SN + 1;

Insert into Inventory (Serial\_No, Aisle\_No, Row\_No, WarehouseID, ProductID)

values(SN,2,3,1,13);

END loop;

-- productID 14

<< loop >>

FOR i IN 1..10 LOOP

SN := SN + 1;

Insert into Inventory (Serial\_No, Aisle\_No, Row\_No, WarehouseID, ProductID)

values(SN,2,4,1,14);

END loop;

-- productID 15

<< loop >>

FOR i IN 1..10 LOOP

SN := SN + 1;

Insert into Inventory (Serial\_No, Aisle\_No, Row\_No, WarehouseID, ProductID)

values(SN,2,5,1,15);

END loop;

-- productID 16

<< loop >>

FOR i IN 1..5 LOOP

SN := SN + 1;

Insert into Inventory (Serial\_No, Aisle\_No, Row\_No, WarehouseID, ProductID)

values(SN,2,6,1,16);

END loop;

-- productID 17

<< loop >>

FOR i IN 1..5 LOOP

SN := SN + 1;

Insert into Inventory (Serial\_No, Aisle\_No, Row\_No, WarehouseID, ProductID)

values(SN,2,7,1,17);

END loop;

-- productID 18

<< loop >>

FOR i IN 1..15 LOOP

SN := SN + 1;

Insert into Inventory (Serial\_No, Aisle\_No, Row\_No, WarehouseID, ProductID)

values(SN,5,1,1,18);

END loop;

-- productID 19

<< loop >>

FOR i IN 1..15 LOOP

SN := SN + 1;

Insert into Inventory (Serial\_No, Aisle\_No, Row\_No, WarehouseID, ProductID)

values(SN,5,2,1,19);

END loop;

-- productID 20

<< loop >>

FOR i IN 1..10 LOOP

SN := SN + 1;

Insert into Inventory (Serial\_No, Aisle\_No, Row\_No, WarehouseID, ProductID)

values(SN,5,3,1,20);

END loop;

-- productID 21

<< loop >>

FOR i IN 1..5 LOOP

SN := SN + 1;

Insert into Inventory (Serial\_No, Aisle\_No, Row\_No, WarehouseID, ProductID)

values(SN,2,8,1,21);

END loop;

-- productID 22

<< loop >>

FOR i IN 1..10 LOOP

SN := SN + 1;

Insert into Inventory (Serial\_No, Aisle\_No, Row\_No, WarehouseID, ProductID)

values(SN,2,9,1,22);

END loop;

-- productID 23

<< loop >>

FOR i IN 1..5 LOOP

SN := SN + 1;

Insert into Inventory (Serial\_No, Aisle\_No, Row\_No, WarehouseID, ProductID)

values(SN,2,10,1,23);

END loop;

-- productID 24

<< loop >>

FOR i IN 1..5 LOOP

SN := SN + 1;

Insert into Inventory (Serial\_No, Aisle\_No, Row\_No, WarehouseID, ProductID)

values(SN,3,1,1,24);

END loop;

-- productID 25

<< loop >>

FOR i IN 1..5 LOOP

SN := SN + 1;

Insert into Inventory (Serial\_No, Aisle\_No, Row\_No, WarehouseID, ProductID)

values(SN,3,2,1,25);

END loop;

-- productID 26

<< loop >>

FOR i IN 1..10 LOOP

SN := SN + 1;

Insert into Inventory (Serial\_No, Aisle\_No, Row\_No, WarehouseID, ProductID)

values(SN,5,4,1,26);

END loop;

END;

/

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- PRICE

DECLARE

i number(3);

BEGIN

-- PRICE 1-10

<< loop >>

FOR i IN 1..10 LOOP

insert into price(price, inventoryid) values(2300,i);

END loop;

-- Inventory 11-20

<< loop >>

FOR i IN 11..20 LOOP

Insert into PRICE (price, inventoryid)

values(2250,i);

END loop;

-- Inventory 21..30

<< loop >>

FOR i IN 21..30 LOOP

Insert into PRICE (price, inventoryid)

values(2350,i);

END loop;

-- Inventory 31..35

<< loop >>

FOR i IN 31..35 LOOP

Insert into PRICE (price, inventoryid)

values(2350,i);

END loop;

-- Inventory 36..40

<< loop >>

FOR i IN 36..40 LOOP

Insert into PRICE (price, inventoryid)

values(2310,i);

END loop;

-- Inventory 41..50

<< loop >>

FOR i IN 41..50 LOOP

Insert into PRICE (price, inventoryid)

values(2000,i);

END loop;

-- Inventory 51..55

<< loop >>

FOR i IN 51..55 LOOP

Insert into PRICE (price, inventoryid)

values(2030,i);

END loop;

-- Inventory 56..65

<< loop >>

FOR i IN 56..65 LOOP

Insert into PRICE (price, inventoryid)

values(2035,i);

END loop;

-- Inventory 66..70

<< loop >>

FOR i IN 66..70 LOOP

Insert into PRICE (price, inventoryid)

values(1950,i);

END loop;

-- Inventory 71..80

<< loop >>

FOR i IN 71..80 LOOP

Insert into PRICE (price, inventoryid)

values(1850,i);

END loop;

-- Inventory 81..100

<< loop >>

FOR i IN 81..100 LOOP

Insert into PRICE (price, inventoryid)

values(3000,i);

END loop;

-- Inventory 101..105

<< loop >>

FOR i IN 101..105 LOOP

Insert into PRICE (price, inventoryid)

values(3300,i);

END loop;

-- Inventory 106..110

<< loop >>

FOR i IN 106..110 LOOP

Insert into PRICE (price, inventoryid)

values(3350,i);

END loop;

-- Inventory 111..120

<< loop >>

FOR i IN 111..120 LOOP

Insert into PRICE (price, inventoryid)

values(3900,i);

END loop;

-- Inventory 121..130

<< loop >>

FOR i IN 121..130 LOOP

Insert into PRICE (price, inventoryid)

values(3950,i);

END loop;

-- Inventory 131..135

<< loop >>

FOR i IN 131..135 LOOP

Insert into PRICE (price, inventoryid)

values(3850,i);

END loop;

-- Inventory 136..140

<< loop >>

FOR i IN 136..140 LOOP

Insert into PRICE (price, inventoryid)

values(3900,i);

END loop;

-- Inventory 141..155

<< loop >>

FOR i IN 141..155 LOOP

Insert into PRICE (price, inventoryid)

values(80,i);

END loop;

-- Inventory 156..170

<< loop >>

FOR i IN 156..170 LOOP

Insert into PRICE (price, inventoryid)

values(120,i);

END loop;

-- Inventory 171..180

<< loop >>

FOR i IN 171..180 LOOP

Insert into PRICE (price, inventoryid)

values(55,i);

END loop;

-- Inventory 181..185

<< loop >>

FOR i IN 181..185 LOOP

Insert into PRICE (price, inventoryid)

values(4500,i);

END loop;

-- Inventory 186..195

<< loop >>

FOR i IN 186..195 LOOP

Insert into PRICE (price, inventoryid)

values(4000,i);

END loop;

-- Inventory 196..200

<< loop >>

FOR i IN 196..200 LOOP

Insert into PRICE (price, inventoryid)

values(4700,i);

END loop;

-- Inventory 201..205

<< loop >>

FOR i IN 201..205 LOOP

Insert into PRICE (price, inventoryid)

values(4700,i);

END loop;

-- Inventory 206..210

<< loop >>

FOR i IN 206..210 LOOP

Insert into PRICE (price, inventoryid)

values(5000,i);

END loop;

-- Inventory 211..220

<< loop >>

FOR i IN 211..220 LOOP

Insert into PRICE (price, inventoryid)

values(20,i);

END loop;

END;

/

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

--Sales\_Transaction

Insert into sales\_transaction (TAMOUNT, CUSTOMERID) VALUES(18920, 1);

Insert into sales\_transaction (TAMOUNT, CUSTOMERID) VALUES(74800, 2);

Insert into sales\_transaction (TAMOUNT, CUSTOMERID) VALUES(153500, 3);

Insert Into Sales\_Transaction (TAMOUNT, CUSTOMERID) VALUES(314400, 4);

Insert Into Sales\_Transaction (TAMOUNT, CUSTOMERID) VALUES(24450, 1);

Insert Into Sales\_Transaction (TAMOUNT, CUSTOMERID) VALUES(187200, 1);

Insert Into Sales\_Transaction (TAMOUNT, CUSTOMERID) VALUES(217075, 2);

Insert Into Sales\_Transaction (TAMOUNT, CUSTOMERID) VALUES(119475, 4);

Insert Into Sales\_Transaction (TAMOUNT, CUSTOMERID) VALUES(191175, 3);

Insert into Sales\_Transaction (TAMOUNT, CUSTOMERID) VALUES(151975, 3);

Insert into Sales\_Transaction (TAMOUNT, CUSTOMERID) VALUES(122750, 3);

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

--Transaction\_Line

-- Customer 1

-- Transaction 1: 3 items

INSERT ALL

into TRANSACTION\_LINE (TID, PRODUCTID, QUANTITY, SALES\_PRICE, Discount) VALUES(1,1,3, 2300, 0)

into TRANSACTION\_LINE (TID, PRODUCTID, QUANTITY, SALES\_PRICE, Discount) VALUES(1,3,5, 2350, 700)

into TRANSACTION\_LINE (TID, PRODUCTID, QUANTITY, SALES\_PRICE, Discount) VALUES(1,9,20, 20, 60)

SELECT \* FROM dual;

-- CUSTOMER 2

-- TRANSACTION 1: 6 ITEMS

INSERT ALL

into TRANSACTION\_LINE (TID, PRODUCTID, QUANTITY, SALES\_PRICE, Discount) VALUES(2,1,8, 2300, 400)

into TRANSACTION\_LINE (TID, PRODUCTID, QUANTITY, SALES\_PRICE, Discount) VALUES(2,2,5, 2250, 700)

into TRANSACTION\_LINE (TID, PRODUCTID, QUANTITY, SALES\_PRICE, Discount) VALUES(2,9,10, 1950, 250)

into TRANSACTION\_LINE (TID, PRODUCTID, QUANTITY, SALES\_PRICE, Discount) VALUES(2,18,10,80, 0)

into TRANSACTION\_LINE (TID, PRODUCTID, QUANTITY, SALES\_PRICE, Discount) VALUES(2,19,10,120, 0)

into TRANSACTION\_LINE (TID, PRODUCTID, QUANTITY, SALES\_PRICE, Discount) VALUES(2,25,5, 5000, 500)

SELECT \* FROM dual;

-- customer 3

-- TRANSACTION 1: 7 ITEMS

INSERT ALL

into TRANSACTION\_LINE (TID, PRODUCTID, QUANTITY, SALES\_PRICE, Discount) VALUES(3, 17, 10, 3900, 500)

into TRANSACTION\_LINE (TID, PRODUCTID, QUANTITY, SALES\_PRICE, Discount) VALUES(3, 18, 50, 80, 400)

into TRANSACTION\_LINE (TID, PRODUCTID, QUANTITY, SALES\_PRICE, Discount) VALUES(3, 19, 30, 120, 0)

into TRANSACTION\_LINE (TID, PRODUCTID, QUANTITY, SALES\_PRICE, Discount) VALUES(3, 21, 5, 4500, 200)

into TRANSACTION\_LINE (TID, PRODUCTID, QUANTITY, SALES\_PRICE, Discount) VALUES(3, 22, 10,4000, 500)

into TRANSACTION\_LINE (TID, PRODUCTID, QUANTITY, SALES\_PRICE, Discount) VALUES(3, 23, 5, 4700, 500)

into TRANSACTION\_LINE (TID, PRODUCTID, QUANTITY, SALES\_PRICE, Discount) VALUES(3, 24, 5, 4700, 500)

SELECT \* FROM dual;

-- customer 4

-- Transaction 1: 5 Items

INSERT ALL

into TRANSACTION\_LINE (TID, PRODUCTID, QUANTITY, SALES\_PRICE, Discount) VALUES(4, 6,20, 2000, 0)

into TRANSACTION\_LINE (TID, PRODUCTID, QUANTITY, SALES\_PRICE, Discount) VALUES(4, 7, 10, 2030, 0)

into TRANSACTION\_LINE (TID, PRODUCTID, QUANTITY, SALES\_PRICE, Discount) VALUES(4, 8, 10, 2035, 0)

into TRANSACTION\_LINE (TID, PRODUCTID, QUANTITY, SALES\_PRICE, Discount) VALUES(4, 9, 15, 1950, 0)

into TRANSACTION\_LINE (TID, PRODUCTID, QUANTITY, SALES\_PRICE, Discount) VALUES(4, 12, 15, 3300, 0)

into TRANSACTION\_LINE (TID, PRODUCTID, QUANTITY, SALES\_PRICE, Discount) VALUES(4, 14, 20, 3900, 0)

into TRANSACTION\_LINE (TID, PRODUCTID, QUANTITY, SALES\_PRICE, Discount) VALUES(4, 16, 20, 3850, 0)

SELECT \* FROM dual;

-- customer 1

-- Transaction 2: 3 items

INSERT ALL

into TRANSACTION\_LINE (TID, PRODUCTID, QUANTITY, SALES\_PRICE, Discount) VALUES(5,1,10, 2300, 100)

into TRANSACTION\_LINE (TID, PRODUCTID, QUANTITY, SALES\_PRICE, Discount)VALUES(5,18,20,80, 50)

into TRANSACTION\_LINE (TID, PRODUCTID, QUANTITY, SALES\_PRICE, Discount) VALUES(5,26,10, 20, 0)

SELECT \* FROM dual;

-- Customer 1

-- Transaction 3: 8 items

INSERT ALL

into TRANSACTION\_LINE (TID,PRODUCTID, QUANTITY, SALES\_PRICE, Discount) VALUES(6, 5, 10, 2310, 0)

into TRANSACTION\_LINE (TID,PRODUCTID, QUANTITY, SALES\_PRICE, Discount) VALUES(6, 6, 10, 2000, 0)

into TRANSACTION\_LINE (TID,PRODUCTID, QUANTITY, SALES\_PRICE, Discount) VALUES(6, 7, 10, 2030, 0)

into TRANSACTION\_LINE (TID,PRODUCTID, QUANTITY, SALES\_PRICE, Discount) VALUES(6, 13,10, 3350, 0)

into TRANSACTION\_LINE (TID,PRODUCTID, QUANTITY, SALES\_PRICE, Discount) VALUES(6, 21, 5, 4500, 100)

into TRANSACTION\_LINE (TID,PRODUCTID, QUANTITY, SALES\_PRICE, Discount) VALUES(6, 22, 5, 4000, 100)

into TRANSACTION\_LINE (TID,PRODUCTID, QUANTITY, SALES\_PRICE, Discount) VALUES(6, 23, 5, 4700, 100)

into TRANSACTION\_LINE (TID,PRODUCTID, QUANTITY, SALES\_PRICE, Discount) VALUES(6, 25, 5, 5000, 400)

SELECT \* FROM dual;

-- Customer 2

-- Transaction 2: 9 items

INSERT ALL

into TRANSACTION\_LINE (TID,PRODUCTID, QUANTITY, SALES\_PRICE, Discount) VALUES(7, 6, 20, 4000, 500)

into TRANSACTION\_LINE (TID,PRODUCTID, QUANTITY, SALES\_PRICE, Discount) VALUES(7, 9, 10, 1950, 0)

into TRANSACTION\_LINE (TID,PRODUCTID, QUANTITY, SALES\_PRICE, Discount) VALUES(7, 11,10, 3000, 0)

into TRANSACTION\_LINE (TID,PRODUCTID, QUANTITY, SALES\_PRICE, Discount) VALUES(7, 17,10, 3900, 200 )

into TRANSACTION\_LINE (TID,PRODUCTID, QUANTITY, SALES\_PRICE, Discount) VALUES(7, 20,5, 55, 0)

into TRANSACTION\_LINE (TID,PRODUCTID, QUANTITY, SALES\_PRICE, Discount) VALUES(7, 21,5, 4500, 100)

into TRANSACTION\_LINE (TID,PRODUCTID, QUANTITY, SALES\_PRICE, Discount) VALUES(7, 22, 5,4000, 100)

into TRANSACTION\_LINE (TID,PRODUCTID, QUANTITY, SALES\_PRICE, Discount) VALUES(7, 23, 5, 4700, 100)

Into TRANSACTION\_LINE (TID,PRODUCTID, QUANTITY, SALES\_PRICE, Discount) VALUES(7, 24, 5, 4700, 200)

SELECT \* FROM dual;

-- Customer 4

-- Transaction 2: 6 items

INSERT ALL

into TRANSACTION\_LINE (TID,PRODUCTID, QUANTITY, SALES\_PRICE, Discount) VALUES(8, 1, 10, 2300,0)

into TRANSACTION\_LINE (TID,PRODUCTID, QUANTITY, SALES\_PRICE, Discount) VALUES(8, 2, 10, 2250,0)

into TRANSACTION\_LINE (TID,PRODUCTID, QUANTITY, SALES\_PRICE, Discount) VALUES(8, 3, 10, 2350,0)

into TRANSACTION\_LINE (TID,PRODUCTID, QUANTITY, SALES\_PRICE, Discount) VALUES(8, 6, 10, 2000,0)

into TRANSACTION\_LINE (TID,PRODUCTID, QUANTITY, SALES\_PRICE, Discount) VALUES(8, 7, 10, 2030,0)

into TRANSACTION\_LINE (TID,PRODUCTID, QUANTITY, SALES\_PRICE, Discount) VALUES(8, 8, 5, 2035,0)

SELECT \* FROM dual;

-- Customer 3

-- Transaction 2: 8 items

INSERT ALL

into TRANSACTION\_LINE (TID,PRODUCTID, QUANTITY, SALES\_PRICE, Discount) VALUES(9, 18, 20, 80,0)

into TRANSACTION\_LINE (TID,PRODUCTID, QUANTITY, SALES\_PRICE, Discount) VALUES(9, 19, 10, 120,0)

into TRANSACTION\_LINE (TID,PRODUCTID, QUANTITY, SALES\_PRICE, Discount) VALUES(9, 20, 5, 55, 0)

into TRANSACTION\_LINE (TID,PRODUCTID, QUANTITY, SALES\_PRICE, Discount) VALUES(9, 21, 10, 4500,0)

into TRANSACTION\_LINE (TID,PRODUCTID, QUANTITY, SALES\_PRICE, Discount) VALUES(9, 23, 10, 4700,200)

into TRANSACTION\_LINE (TID,PRODUCTID, QUANTITY, SALES\_PRICE, Discount) VALUES(9, 24, 10, 4700,200)

into TRANSACTION\_LINE (TID,PRODUCTID, QUANTITY, SALES\_PRICE, Discount) VALUES(9, 25, 10, 5000,500)

into TRANSACTION\_LINE (TID,PRODUCTID, QUANTITY, SALES\_PRICE, Discount) VALUES(9, 26, 10, 20, 0)

SELECT \* FROM dual;

-- Customer 3

-- Transaction 3: 4 items

INSERT ALL

into TRANSACTION\_LINE (TID,PRODUCTID, QUANTITY, SALES\_PRICE, Discount) VALUES(10,1, 10, 2300,200)

into TRANSACTION\_LINE (TID,PRODUCTID, QUANTITY, SALES\_PRICE, Discount) VALUES(10,2, 10,2250,200)

into TRANSACTION\_LINE (TID,PRODUCTID, QUANTITY, SALES\_PRICE, Discount) VALUES(10,3, 10,2350,200)

into TRANSACTION\_LINE (TID,PRODUCTID, QUANTITY, SALES\_PRICE, Discount) VALUES(10,4, 10,2350,200)

into TRANSACTION\_LINE (TID,PRODUCTID, QUANTITY, SALES\_PRICE, Discount) VALUES(10,5, 5,2310,100)

into TRANSACTION\_LINE (TID,PRODUCTID, QUANTITY, SALES\_PRICE, Discount) VALUES(10,6, 5,2000,100)

into TRANSACTION\_LINE (TID,PRODUCTID, QUANTITY, SALES\_PRICE, Discount) VALUES(10,7, 5,2030,100)

into TRANSACTION\_LINE (TID,PRODUCTID, QUANTITY, SALES\_PRICE, Discount) VALUES(10,8, 5,2035,100)

into TRANSACTION\_LINE (TID,PRODUCTID, QUANTITY, SALES\_PRICE, Discount) VALUES(10,9, 5,1950,100)

into TRANSACTION\_LINE (TID,PRODUCTID, QUANTITY, SALES\_PRICE, Discount) VALUES(10,10,5,1850,100)

SELECT \* FROM dual;

-- Customer 3

-- Transaction 4: 7 items

INSERT ALL

into TRANSACTION\_LINE (TID,PRODUCTID, QUANTITY, SALES\_PRICE, Discount) VALUES(11,11, 5, 3000,500)

into TRANSACTION\_LINE (TID,PRODUCTID, QUANTITY, SALES\_PRICE, Discount) VALUES(11,12, 5, 3300,500)

into TRANSACTION\_LINE (TID,PRODUCTID, QUANTITY, SALES\_PRICE, Discount) VALUES(11,13, 5, 3350,500)

into TRANSACTION\_LINE (TID,PRODUCTID, QUANTITY, SALES\_PRICE, Discount) VALUES(11,14, 5, 3900,500)

into TRANSACTION\_LINE (TID,PRODUCTID, QUANTITY, SALES\_PRICE, Discount) VALUES(11,15, 5, 3950,500)

into TRANSACTION\_LINE (TID,PRODUCTID, QUANTITY, SALES\_PRICE, Discount) VALUES(11,16, 5, 3850,500)

into TRANSACTION\_LINE (TID,PRODUCTID, QUANTITY, SALES\_PRICE, Discount) VALUES(11,17, 5, 3900,500)

SELECT \* FROM dual;

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

DECLARE

AR\_DATE\_DUE ACCOUNT\_RECEIVABLE.DATE\_DUE%type;

BEGIN

SELECT TO\_CHAR(SYSDATE, 'MM-DD-YYYY') INTO AR\_DATE\_DUE from dual;

AR\_DATE\_DUE := AR\_DATE\_DUE + 30;

-- TR 1

Insert into ACCOUNT\_RECEIVABLE(TID, AMOUNT, DATE\_DUE, BALANCE\_DUE)

VALUES(1, 18920, AR\_DATE\_DUE, 18920);

-- TR 2

Insert into ACCOUNT\_RECEIVABLE(TID, AMOUNT, DATE\_DUE, BALANCE\_DUE)

VALUES(2, 74300, AR\_DATE\_DUE, 74300);

-- TR 3

Insert into ACCOUNT\_RECEIVABLE(TID, AMOUNT, DATE\_DUE, BALANCE\_DUE)

VALUES(3, 153500, AR\_DATE\_DUE,153500);

-- TR 4

Insert into ACCOUNT\_RECEIVABLE(TID, AMOUNT, DATE\_DUE, BALANCE\_DUE)

VALUES(4, 314400, AR\_DATE\_DUE,157200);

Insert into ACCOUNT\_RECEIVABLE(TID, AMOUNT, DATE\_DUE, BALANCE\_DUE)

VALUES(4, 157200, AR\_DATE\_DUE+30,157200);

-- TR 5

Insert into ACCOUNT\_RECEIVABLE(TID, AMOUNT, DATE\_DUE, BALANCE\_DUE)

VALUES(5, 24450, AR\_DATE\_DUE,24450);

-- TR 6

Insert into ACCOUNT\_RECEIVABLE(TID, AMOUNT, DATE\_DUE, BALANCE\_DUE)

VALUES(6, 187200, AR\_DATE\_DUE,187200);

-- TR 7

Insert into ACCOUNT\_RECEIVABLE(TID, AMOUNT, DATE\_DUE, BALANCE\_DUE)

VALUES(7, 257075, AR\_DATE\_DUE,108537.5);

Insert into ACCOUNT\_RECEIVABLE(TID, AMOUNT, DATE\_DUE, BALANCE\_DUE)

VALUES(7, 128537.5, AR\_DATE\_DUE+30,128537.5);

-- TR 8

Insert into ACCOUNT\_RECEIVABLE(TID, AMOUNT, DATE\_DUE, BALANCE\_DUE)

VALUES(8, 119475, AR\_DATE\_DUE, 119475);

-- TR 9

Insert into ACCOUNT\_RECEIVABLE(TID, AMOUNT, DATE\_DUE, BALANCE\_DUE)

VALUES(9, 191375, AR\_DATE\_DUE, 191375);

-- TR 10

Insert into ACCOUNT\_RECEIVABLE(TID, AMOUNT, DATE\_DUE, BALANCE\_DUE)

VALUES(10, 151975, AR\_DATE\_DUE, 151975);

-- TR 11

Insert into ACCOUNT\_RECEIVABLE(TID, AMOUNT, DATE\_DUE, BALANCE\_DUE)

VALUES(11, 122750, AR\_DATE\_DUE, 122750);

END;

/

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

**Data Deletion/Drop SQL Code:**

-- Sales inventory Database DROP.

-- 16 Tables

-- 15 Sequences

-- Date Created: 12/26/2022

-- Date Updated: Null

DROP TABLE Transaction\_Line;

DROP TABLE Price;

DROP TABLE Inventory;

DROP TABLE Product\_Built;

DROP TABLE Model\_Text;

DROP TABLE Product\_Model;

DROP TABLE CATEGORY;

DROP TABLE Product\_Color;

DROP TABLE Product\_Size;

DROP TABLE Warehouse;

DROP TABLE Account\_Receivable;

DROP TABLE SALES\_TRANSACTION;

DROP TABLE Address;

DROP TABLE Address\_Type;

DROP TABLE CUSTOMER;

DROP TABLE REGION;

DROP SEQUENCE INVENTORY\_seq;

DROP SEQUENCE PRODUCT\_BUILT\_seq;

DROP SEQUENCE MODEL\_TEXT\_seq;

DROP SEQUENCE PRODUCT\_MODEL\_seq;

DROP SEQUENCE CATEGORY\_SEQ;

DROP SEQUENCE PRICE\_seq;

DROP SEQUENCE PRODUCT\_COLOR\_seq;

DROP SEQUENCE PRODUCT\_SIZE\_seq;

DROP SEQUENCE Warehouse\_seq;

DROP SEQUENCE Account\_Receivable\_seq;

DROP SEQUENCE SALES\_TRANSACTION\_seq;

DROP SEQUENCE Address\_seq;

DROP SEQUENCE ADDRESS\_TYPE\_seq;

DROP SEQUENCE CUSTOMER\_SEQ;

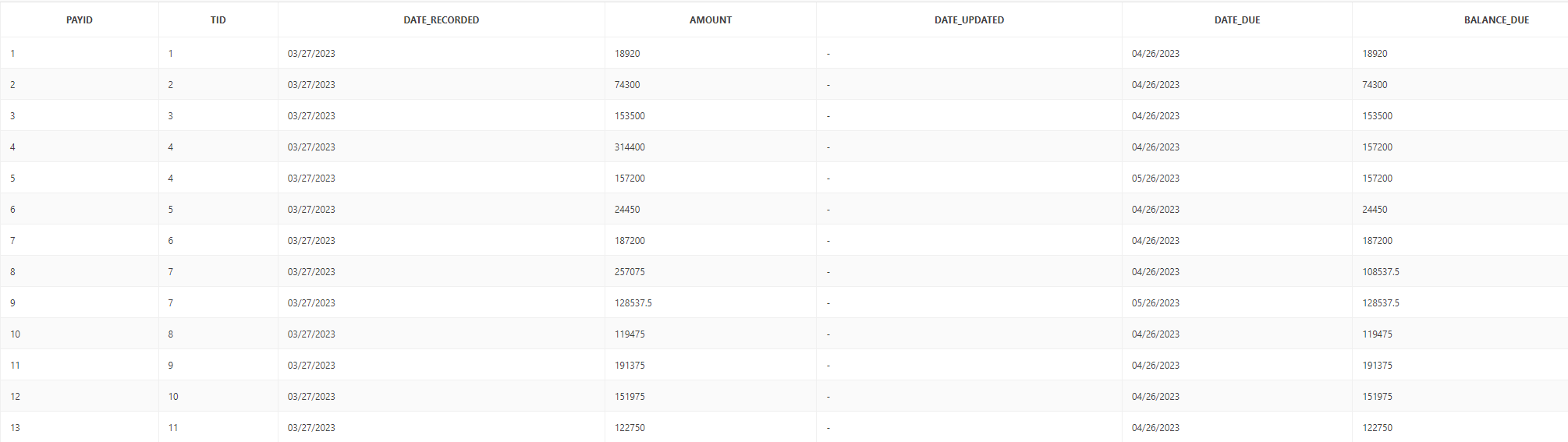
DROP SEQUENCE REGION\_seq;

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

**Sample Data:**

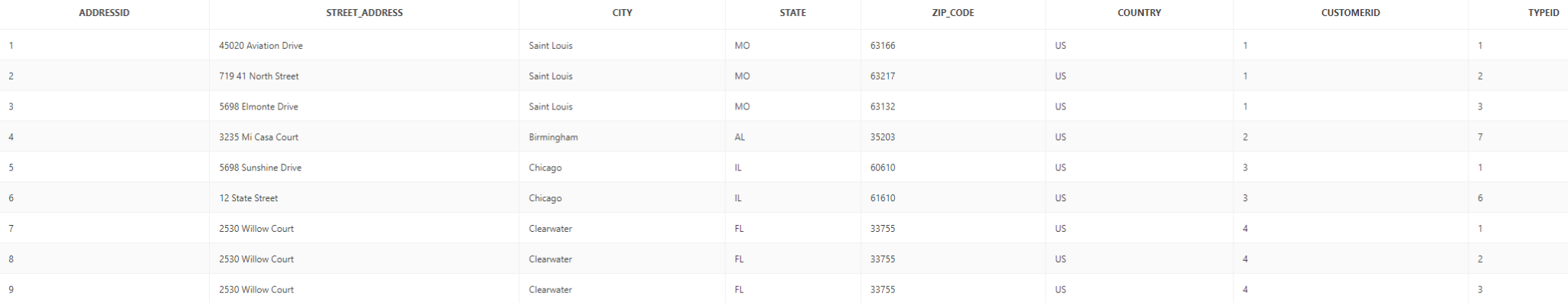
ACCOUNT\_RECEIVABLE





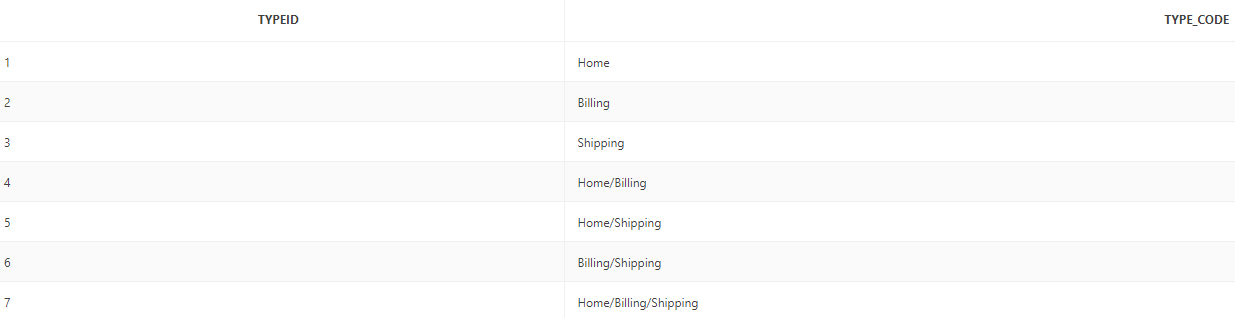
ADDRESS





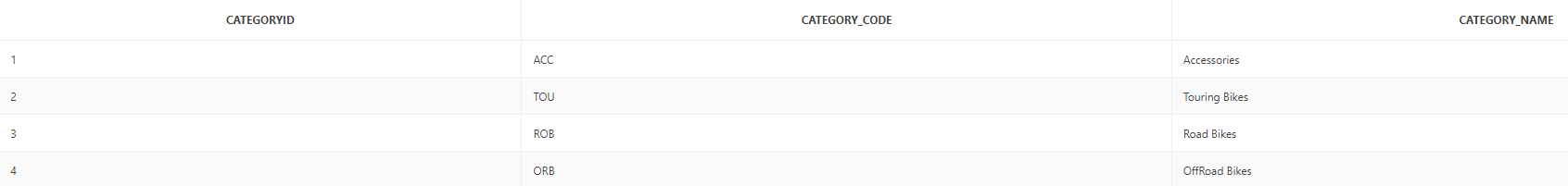
ADDRESS\_TYPE





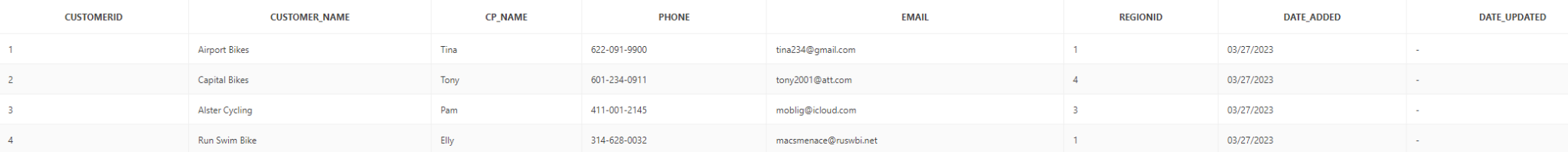
CATEGORY





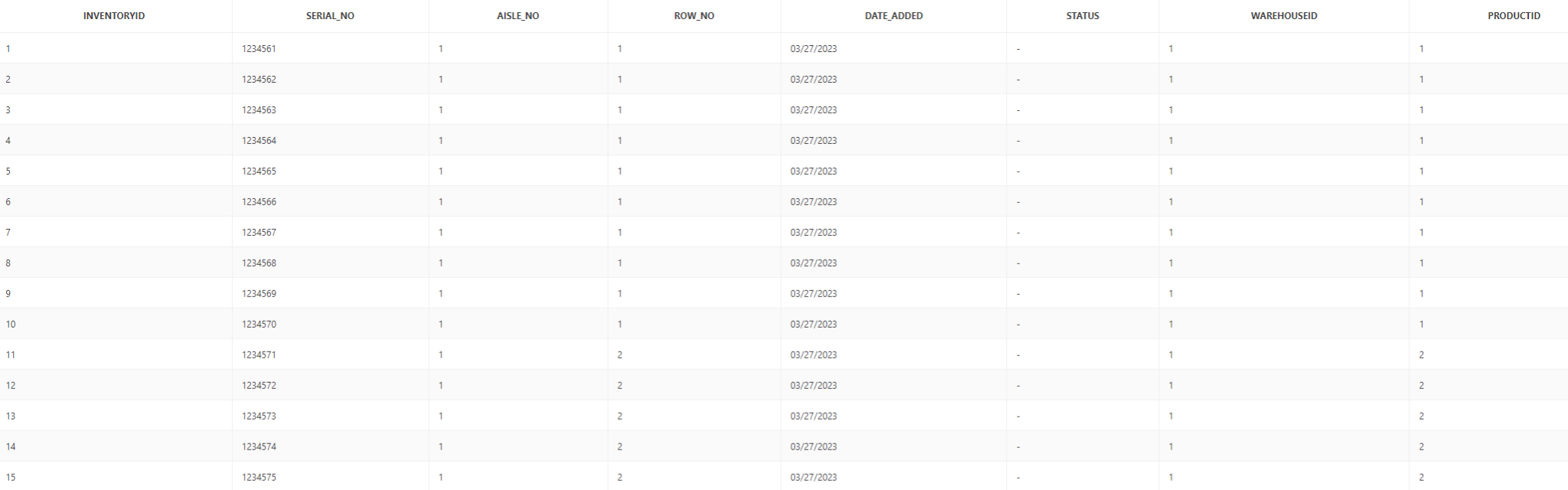
CUSTOMER

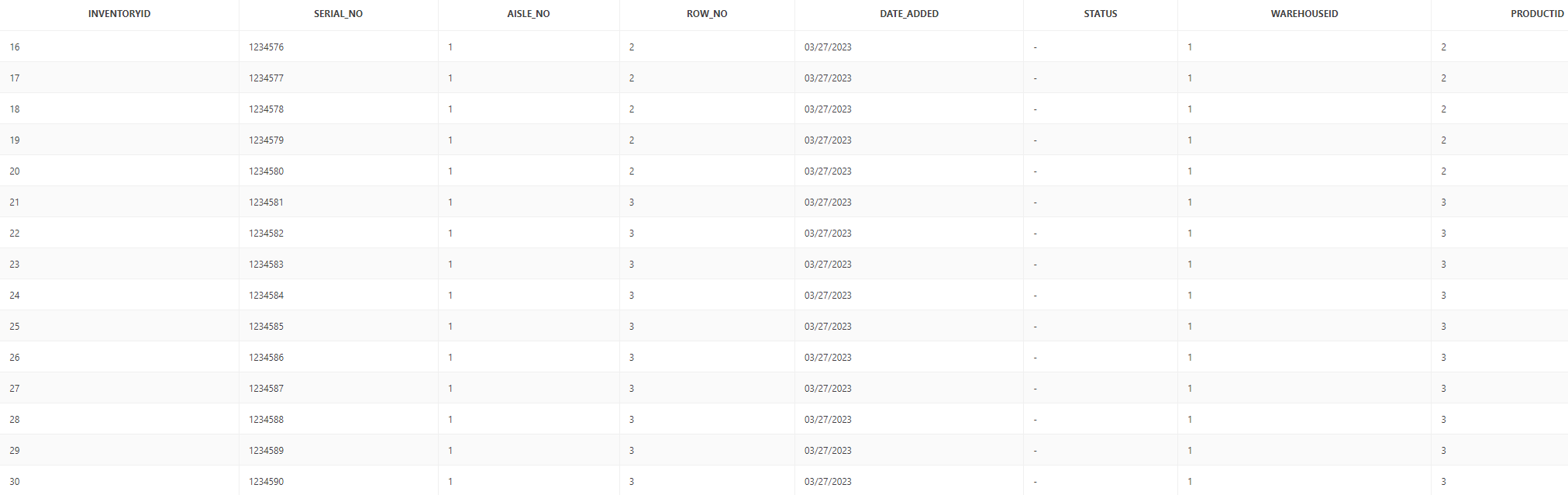


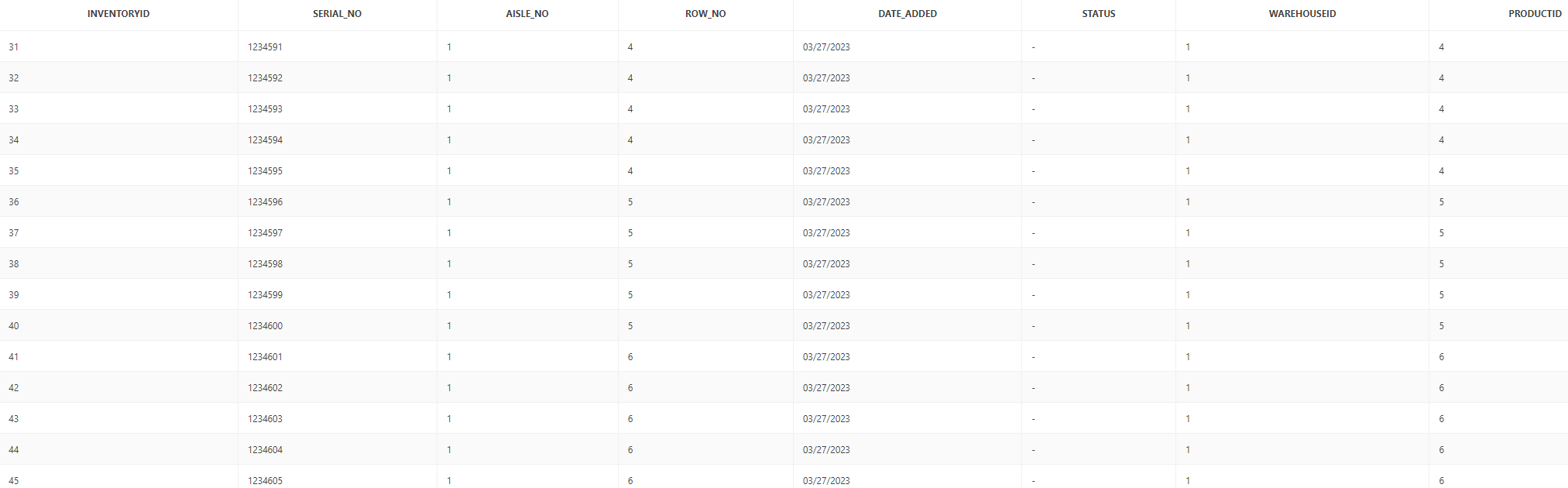


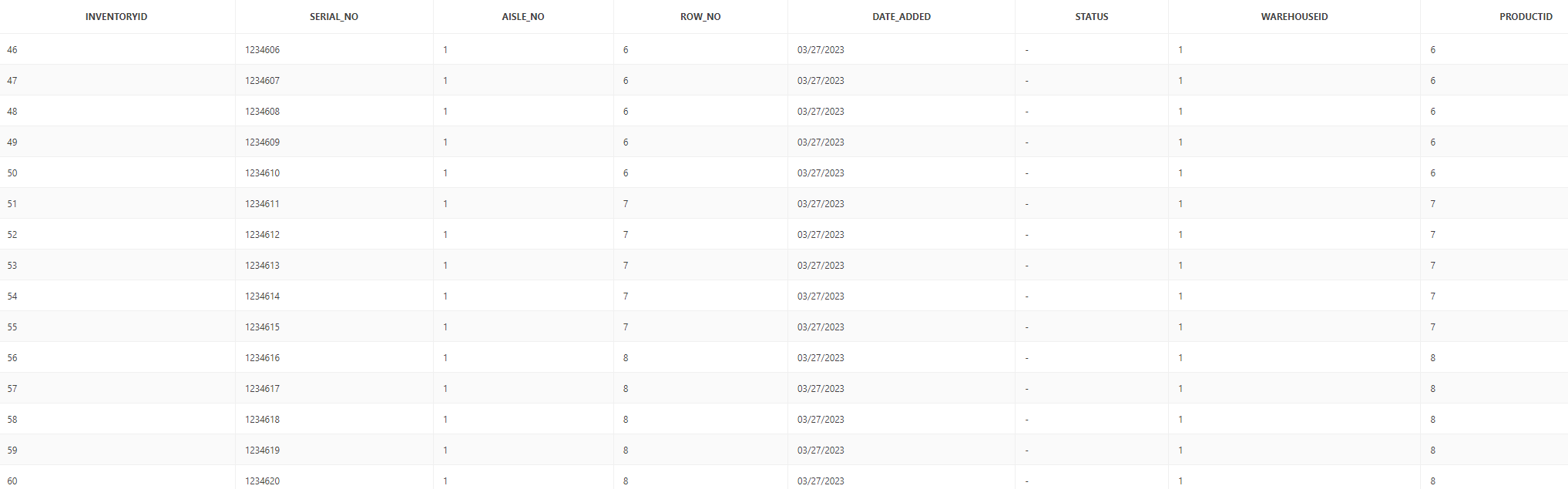
INVENTORY

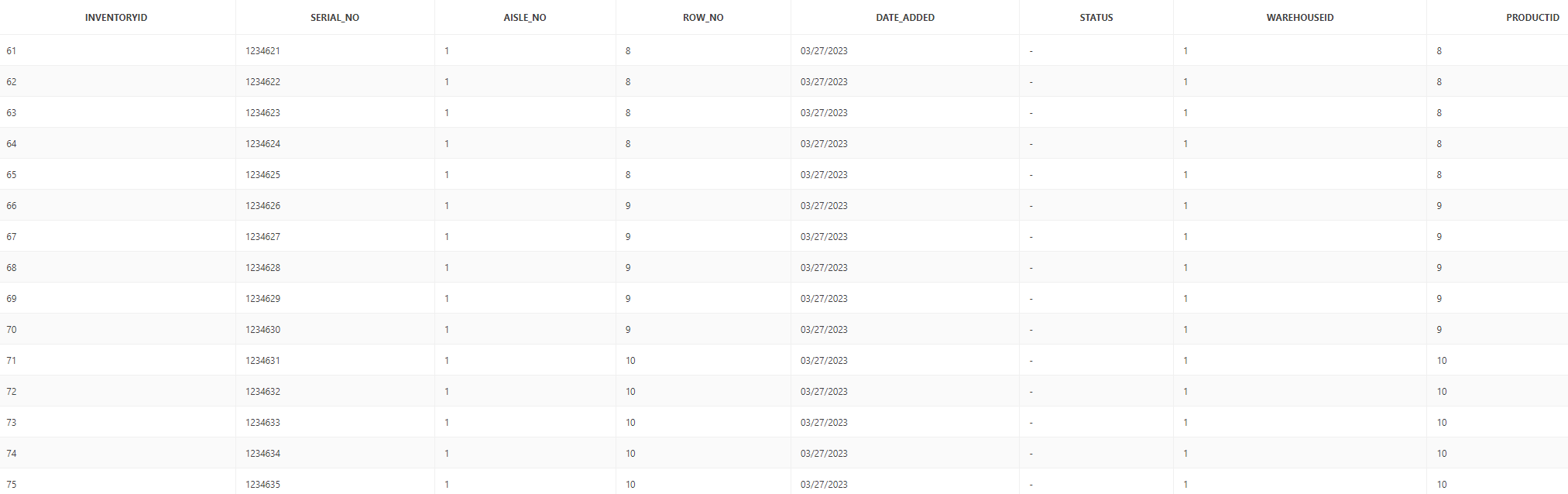


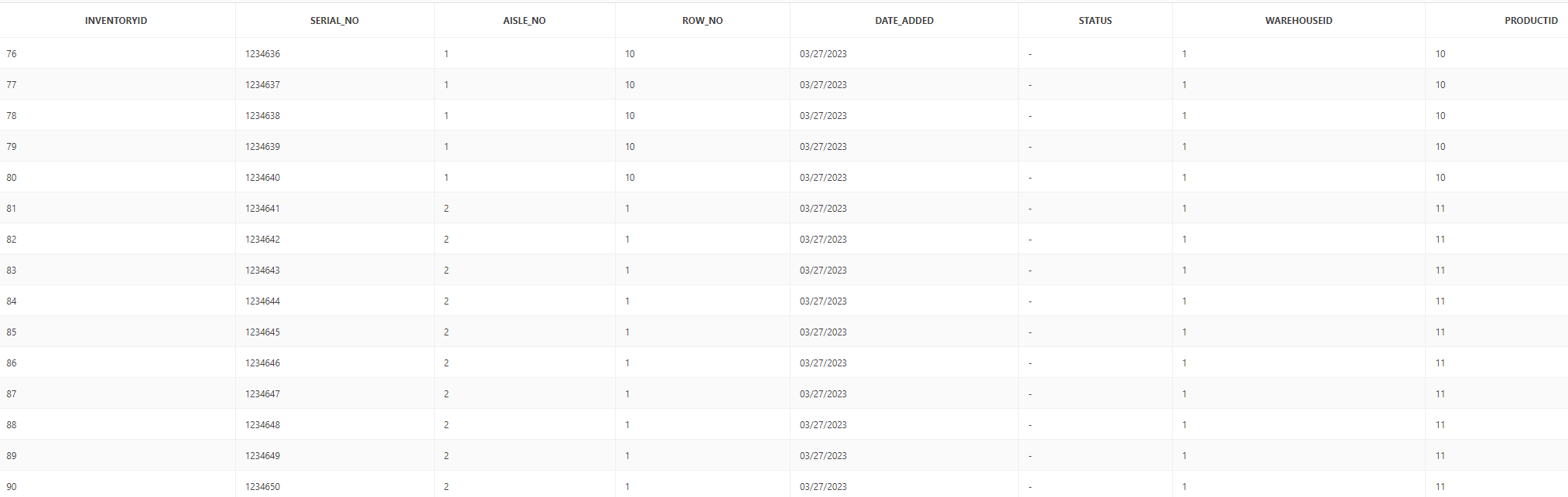


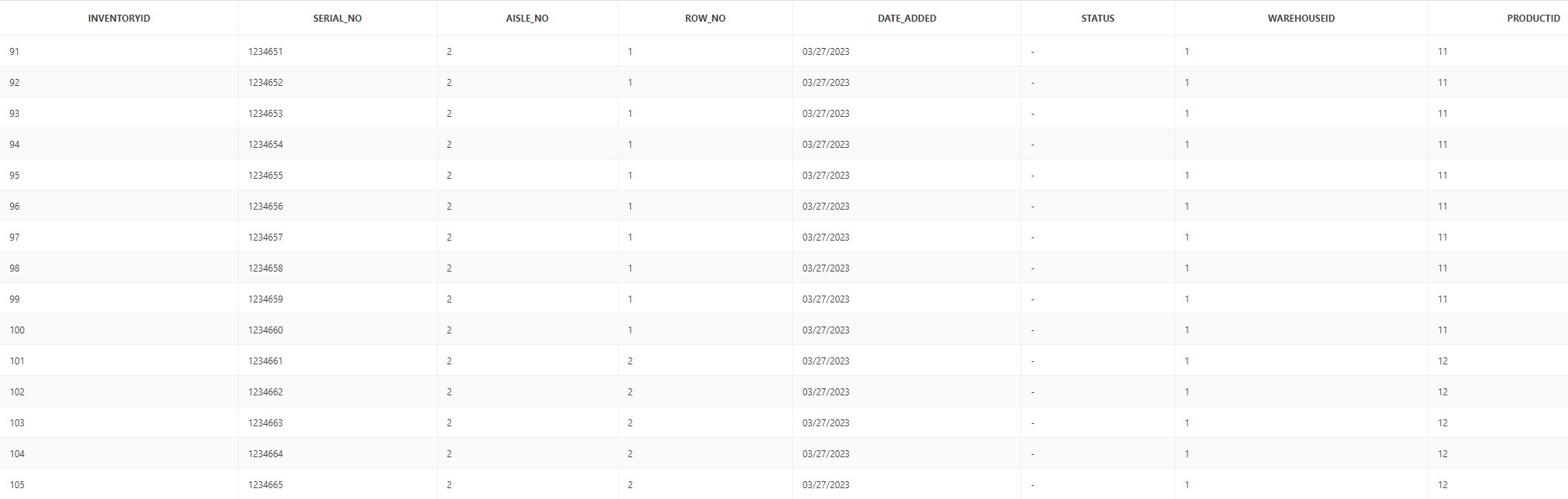


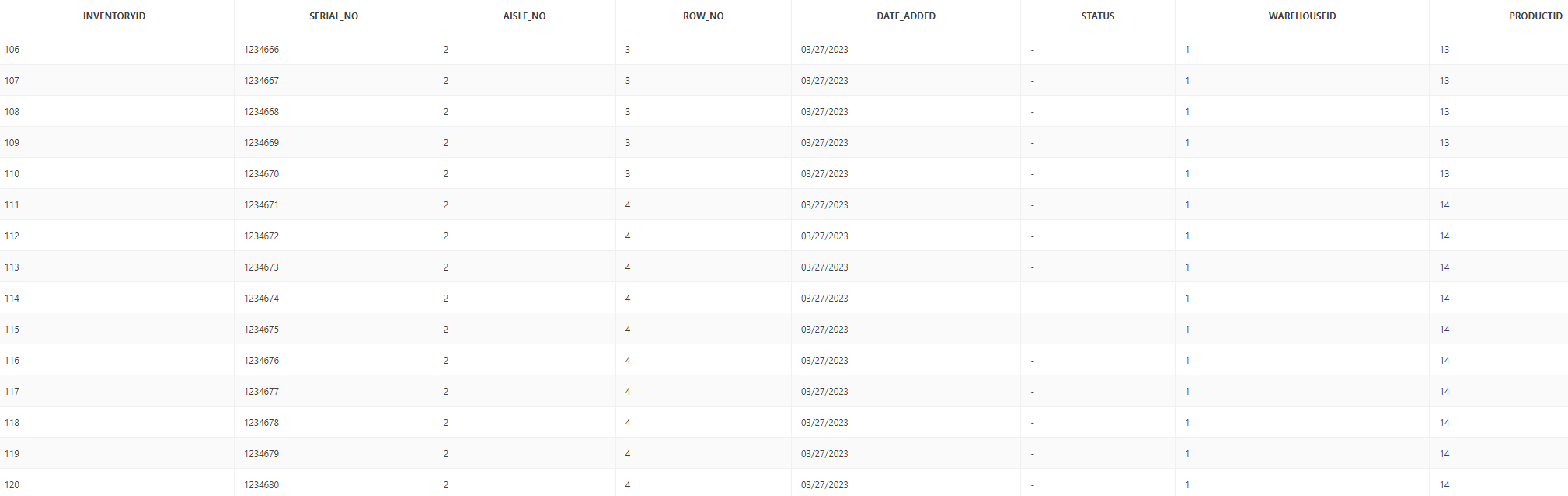


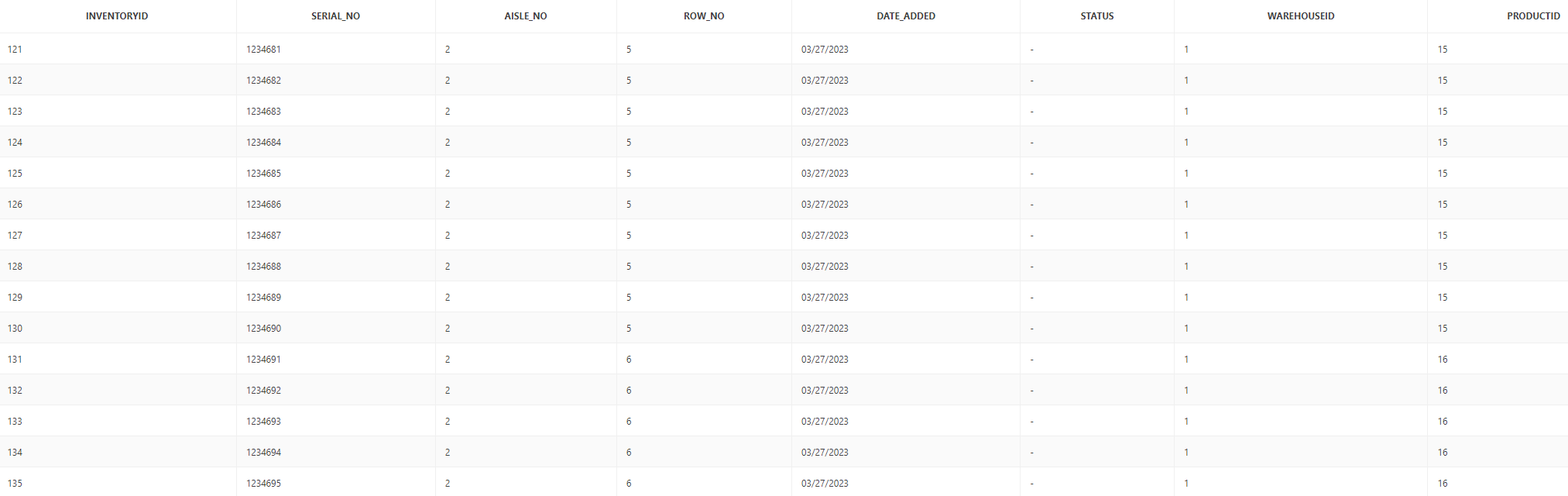


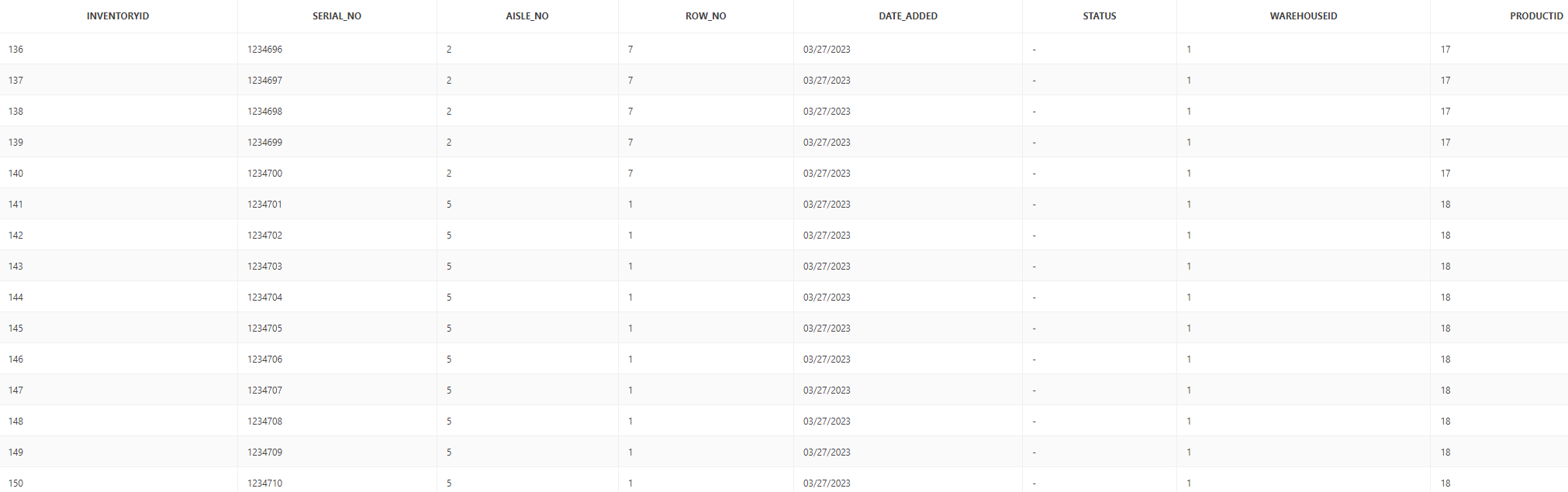


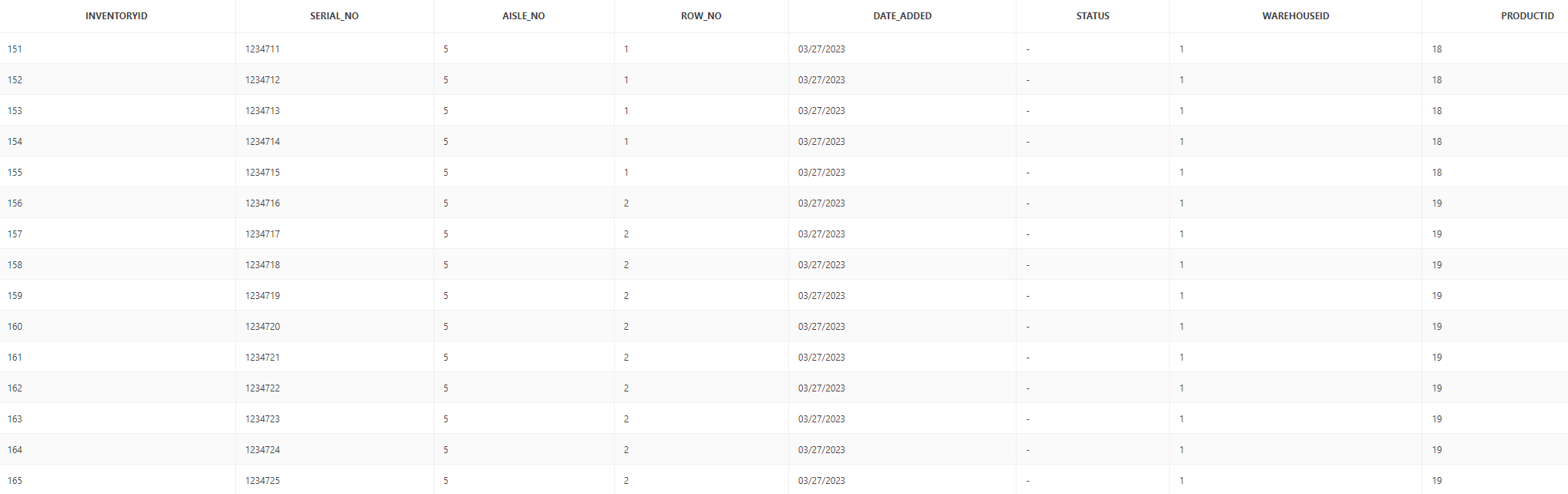


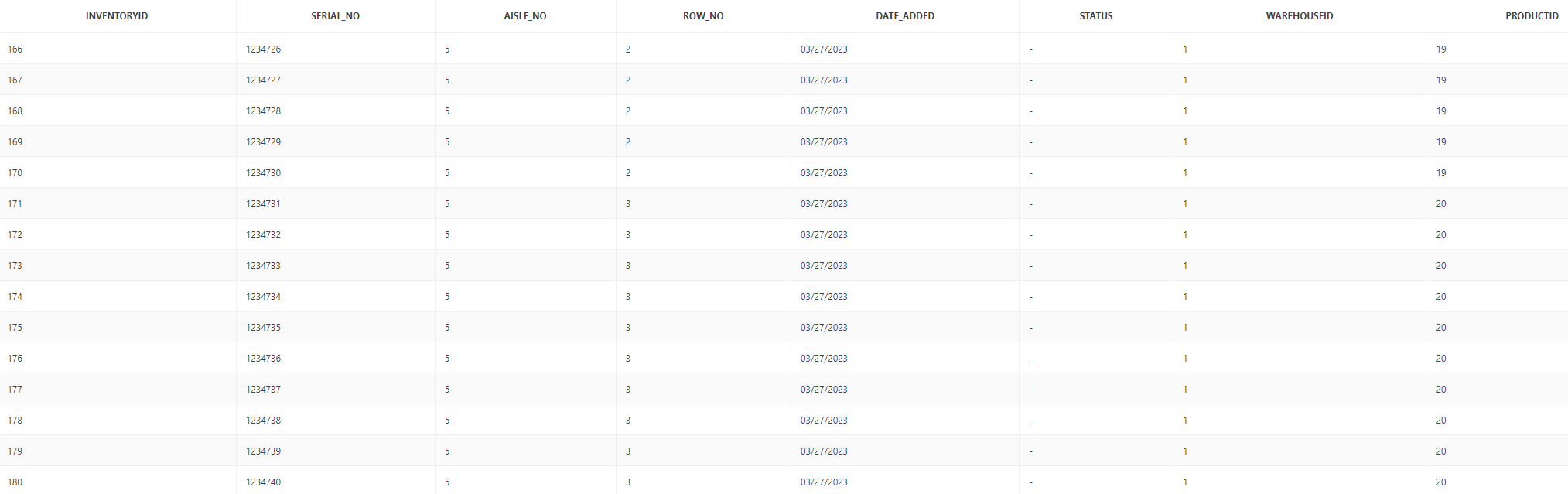


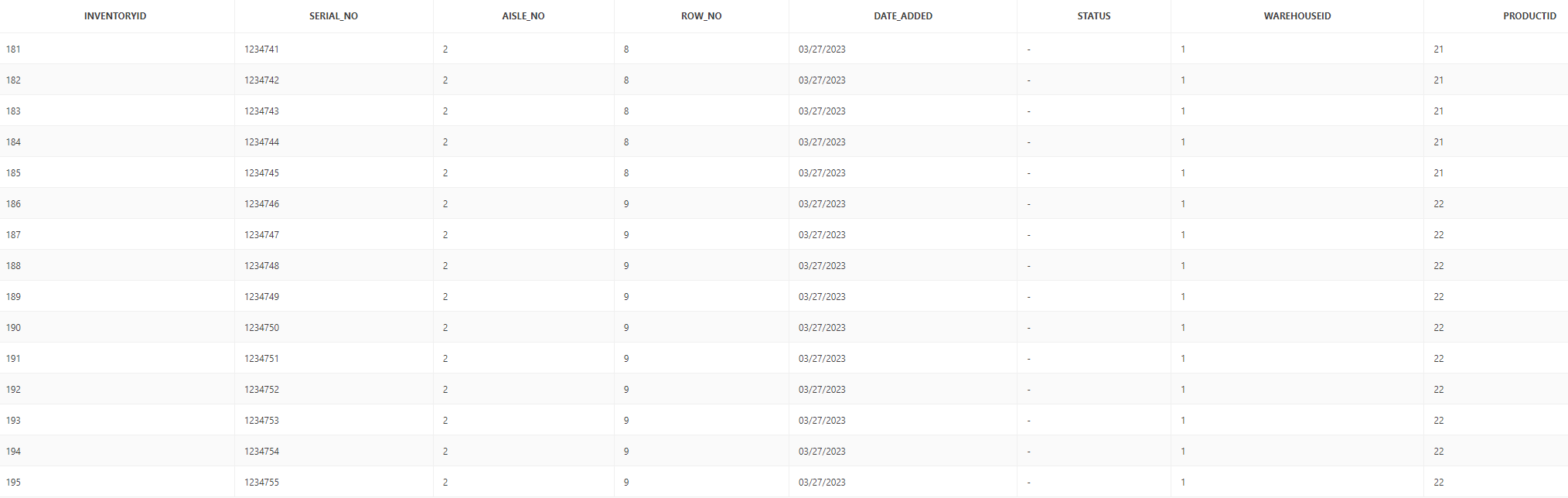


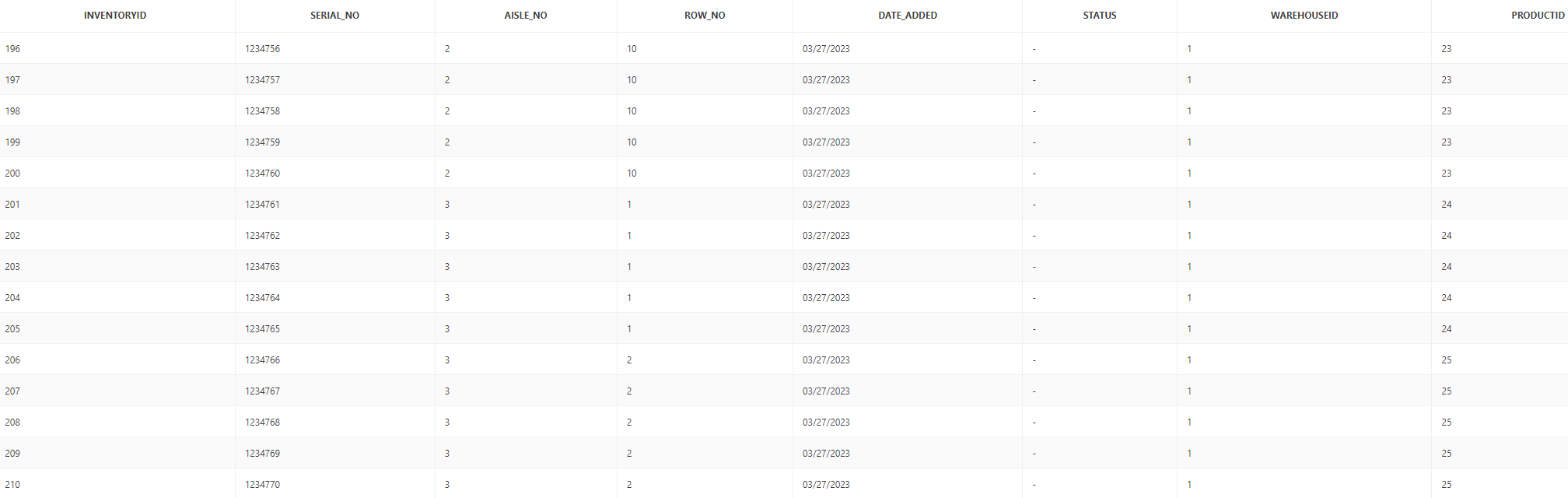


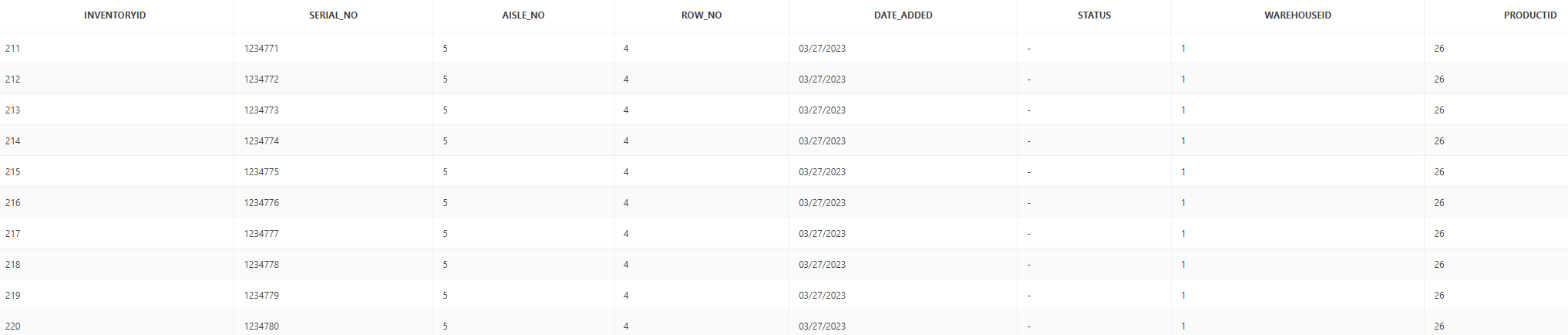




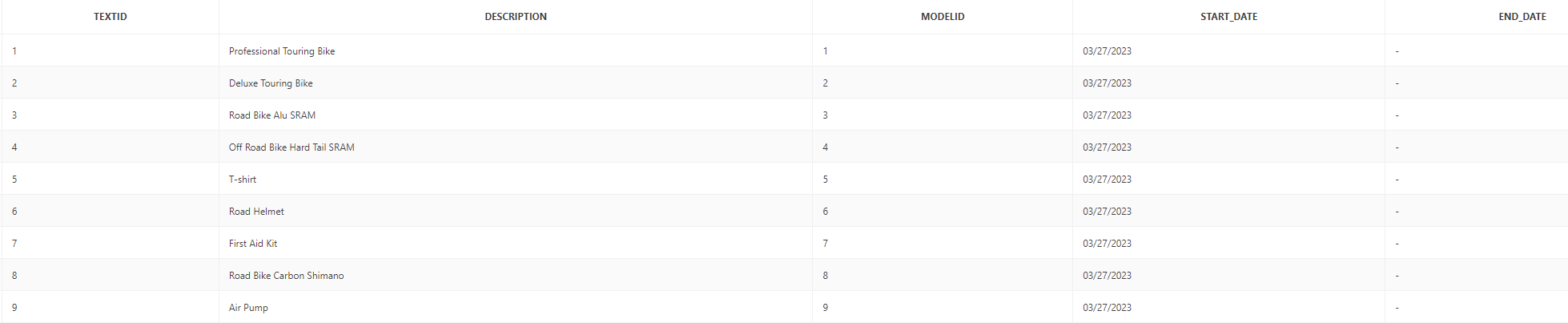






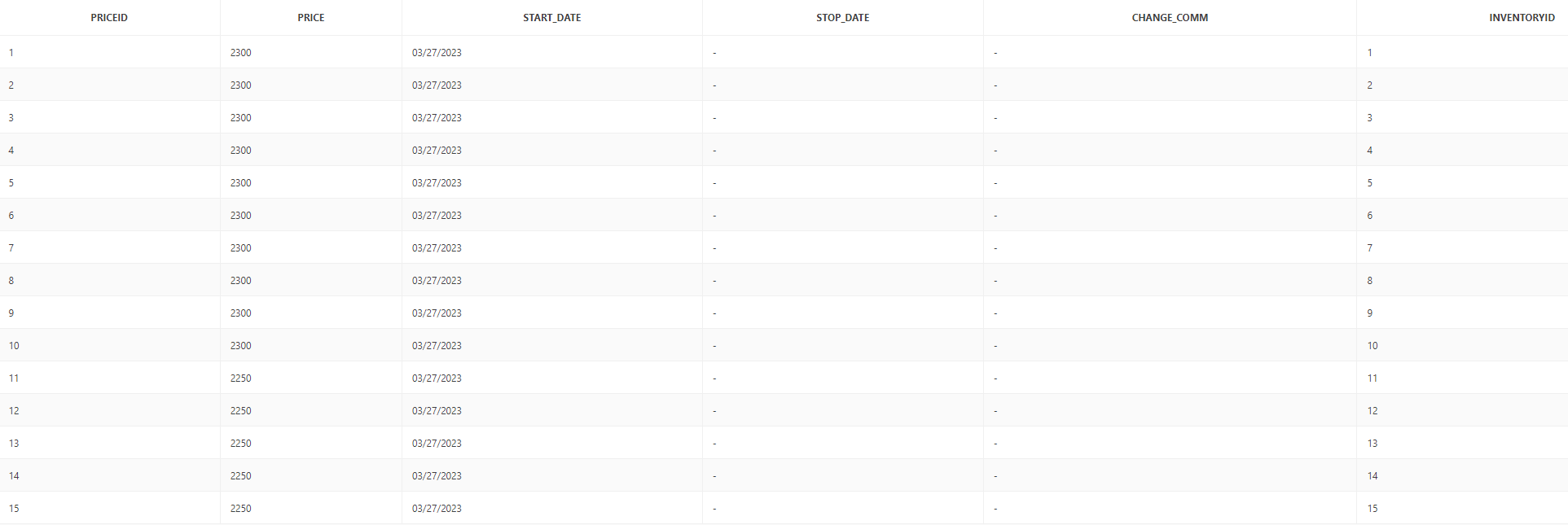


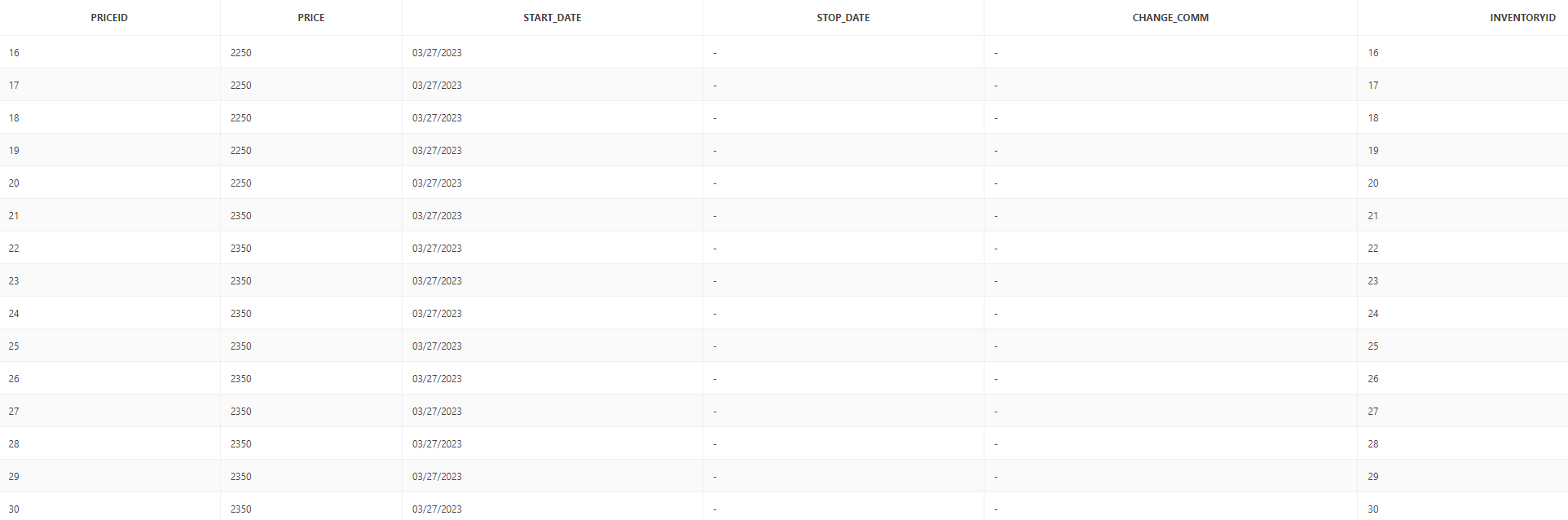
MODEL\_TEXT

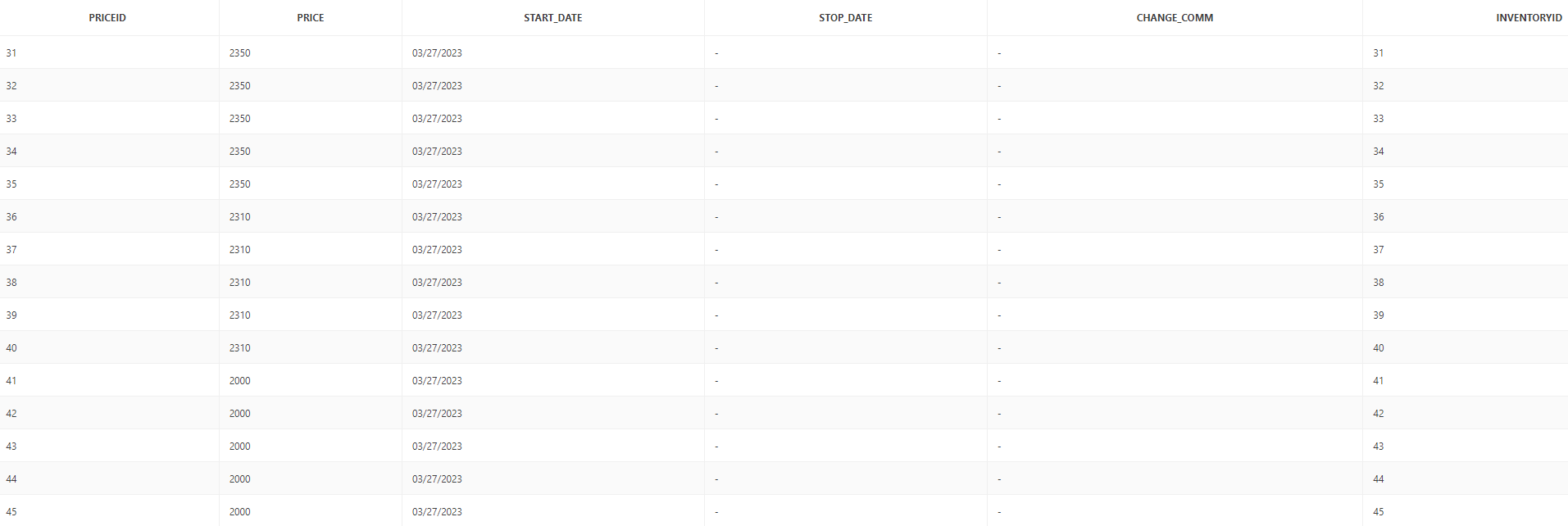


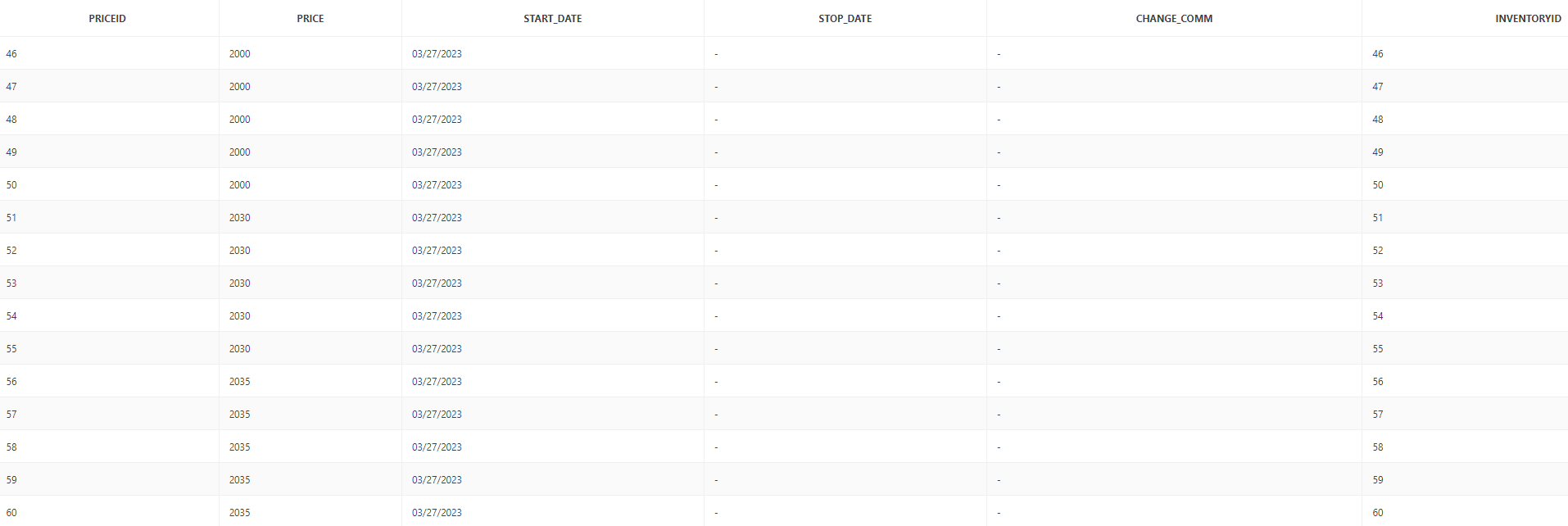
PRICE

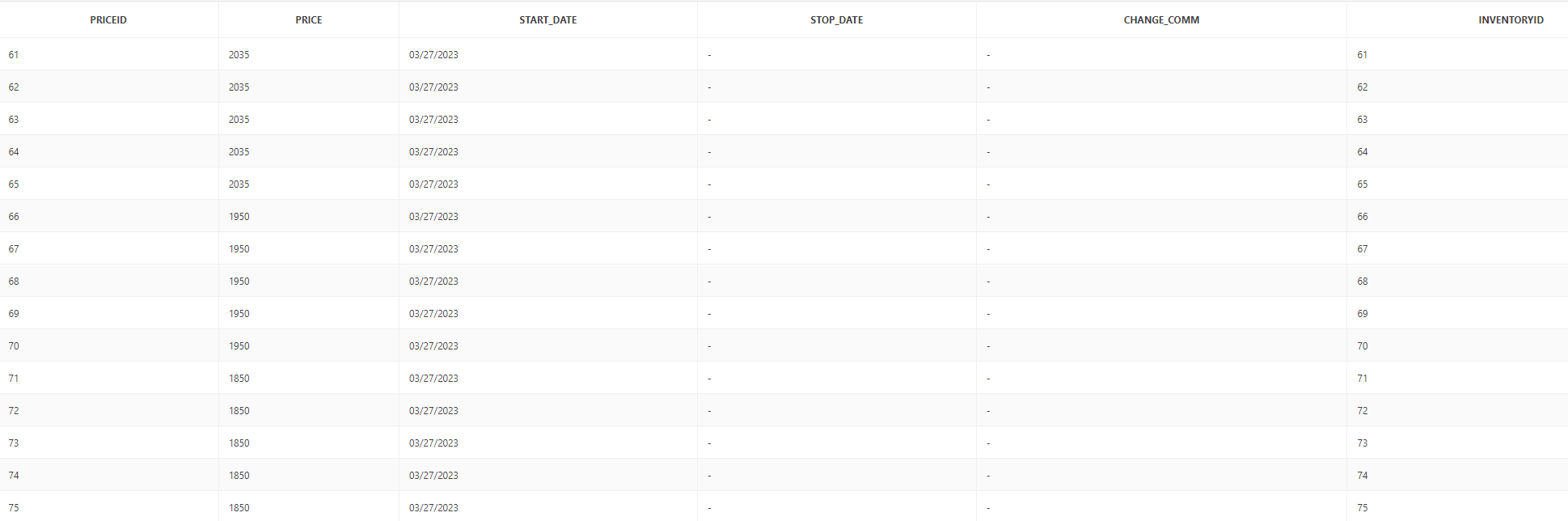


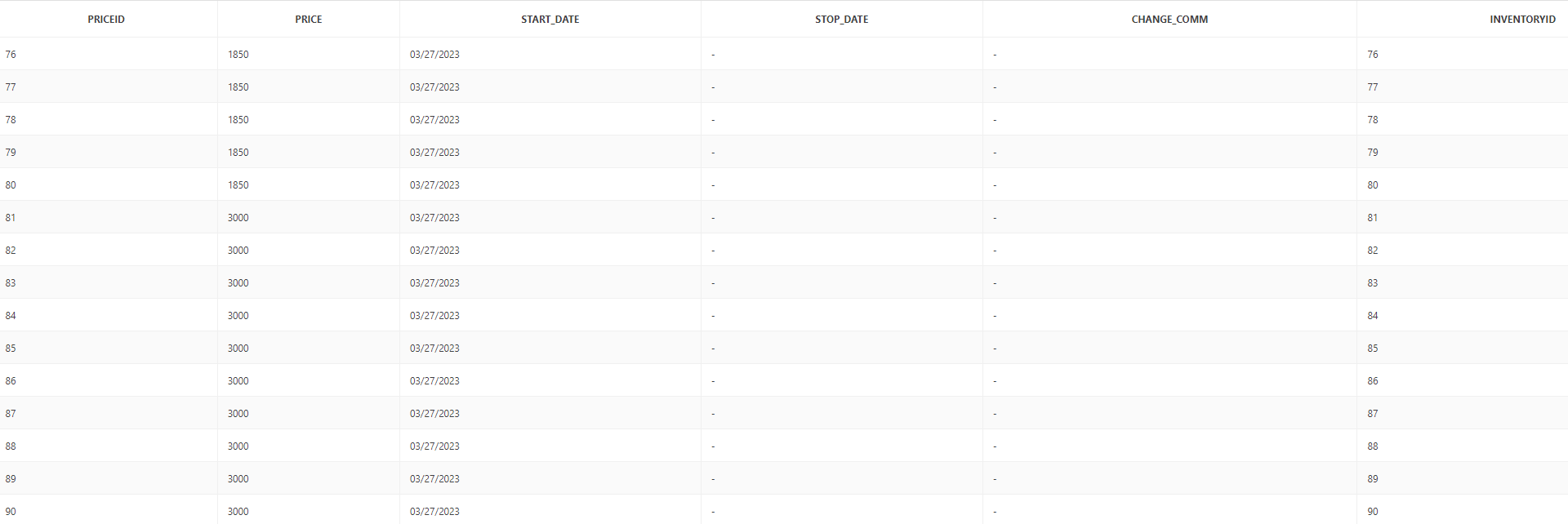


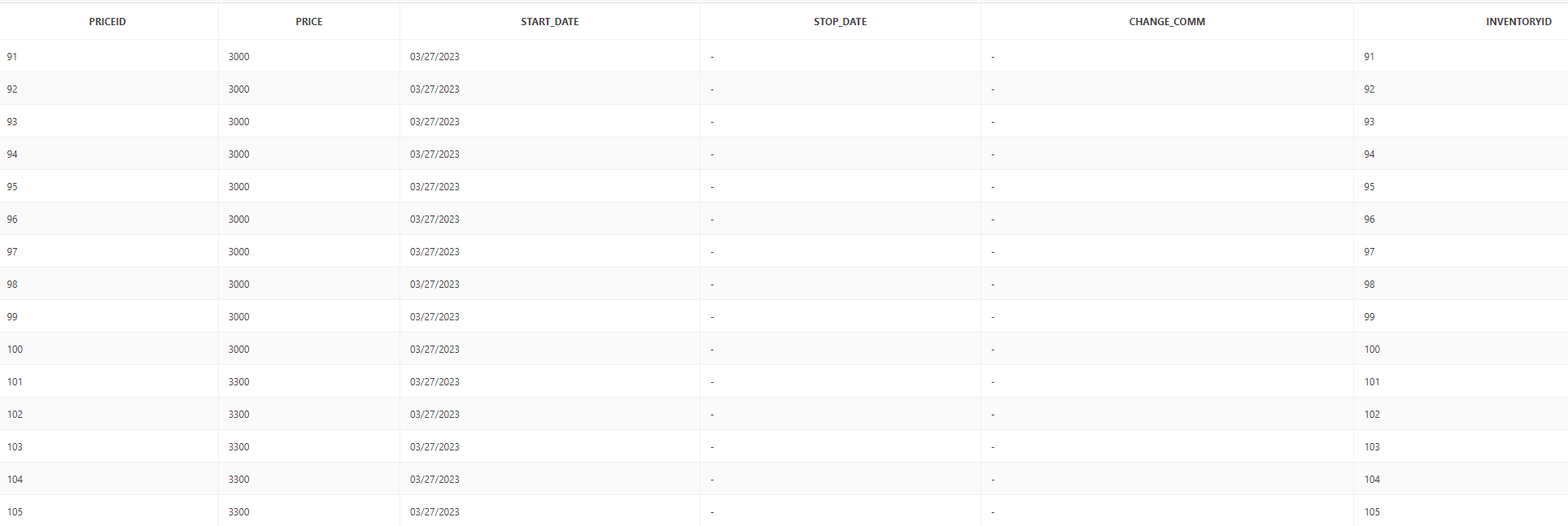


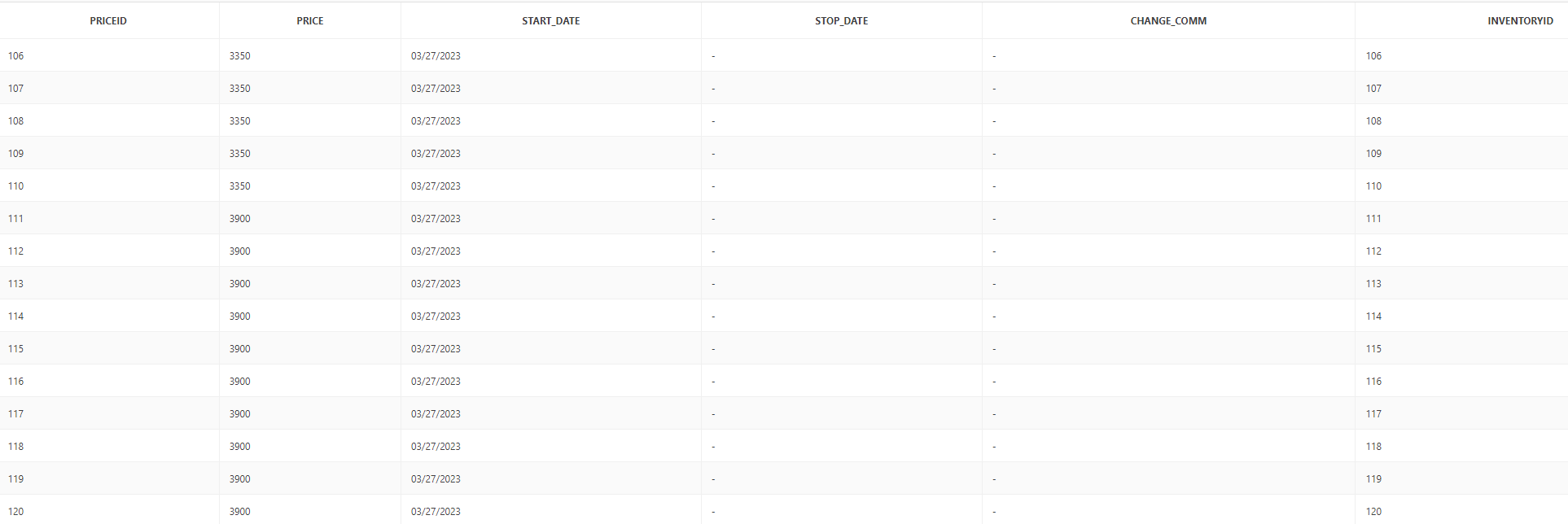


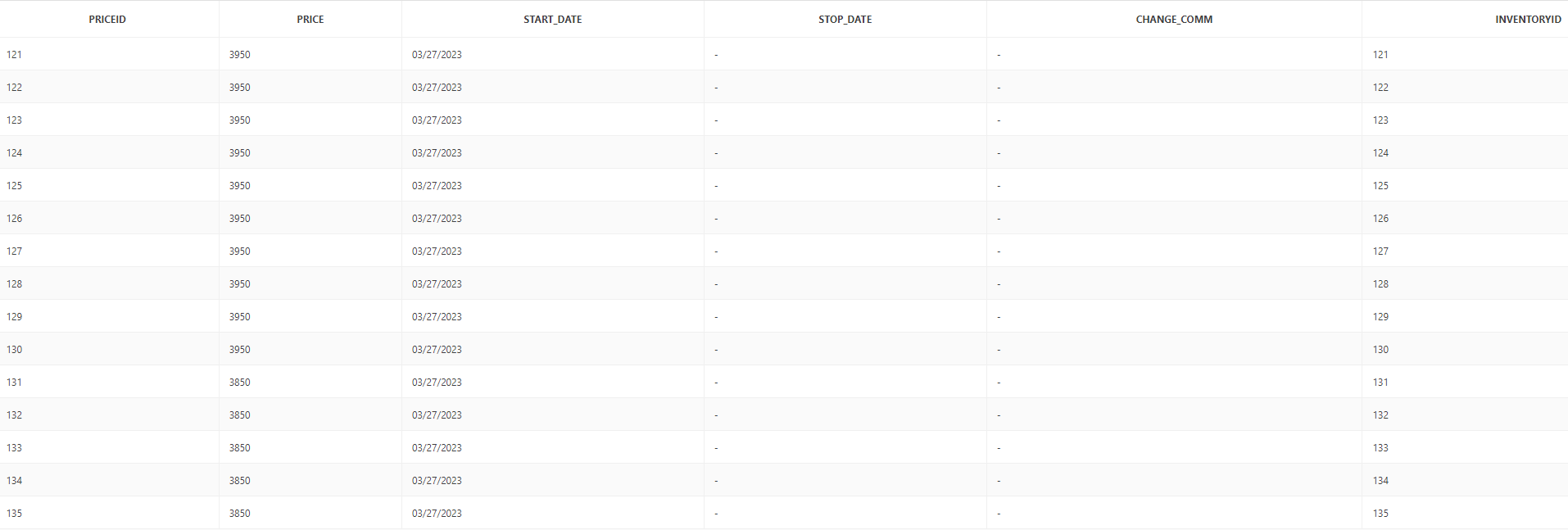


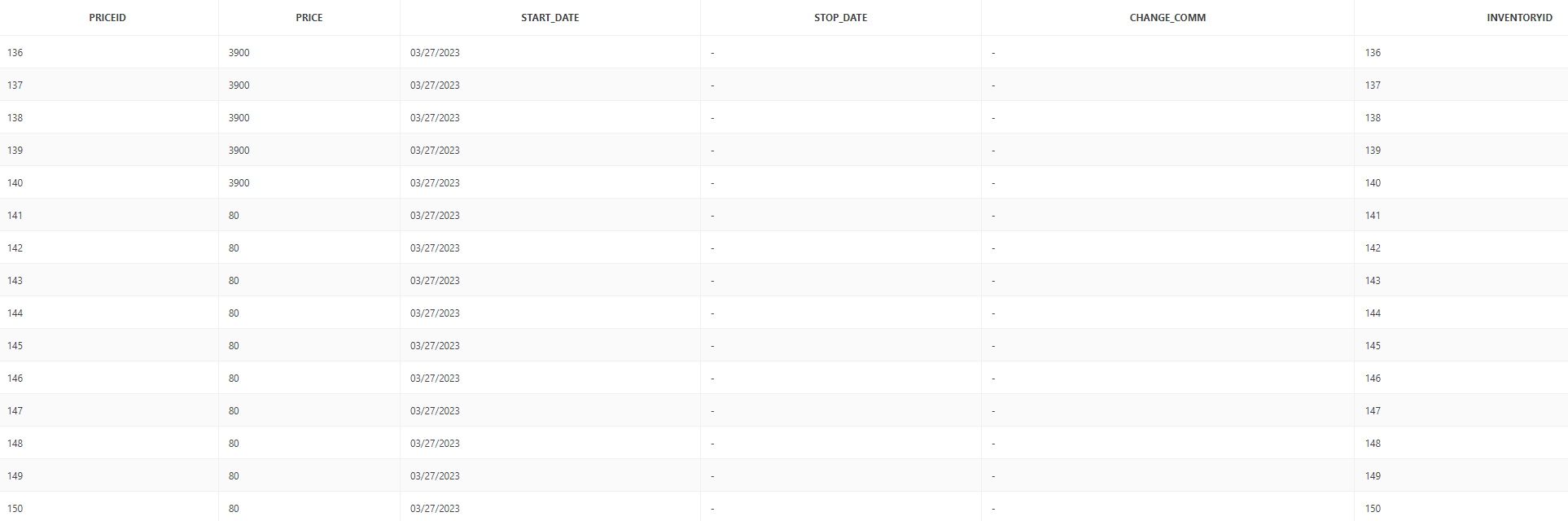


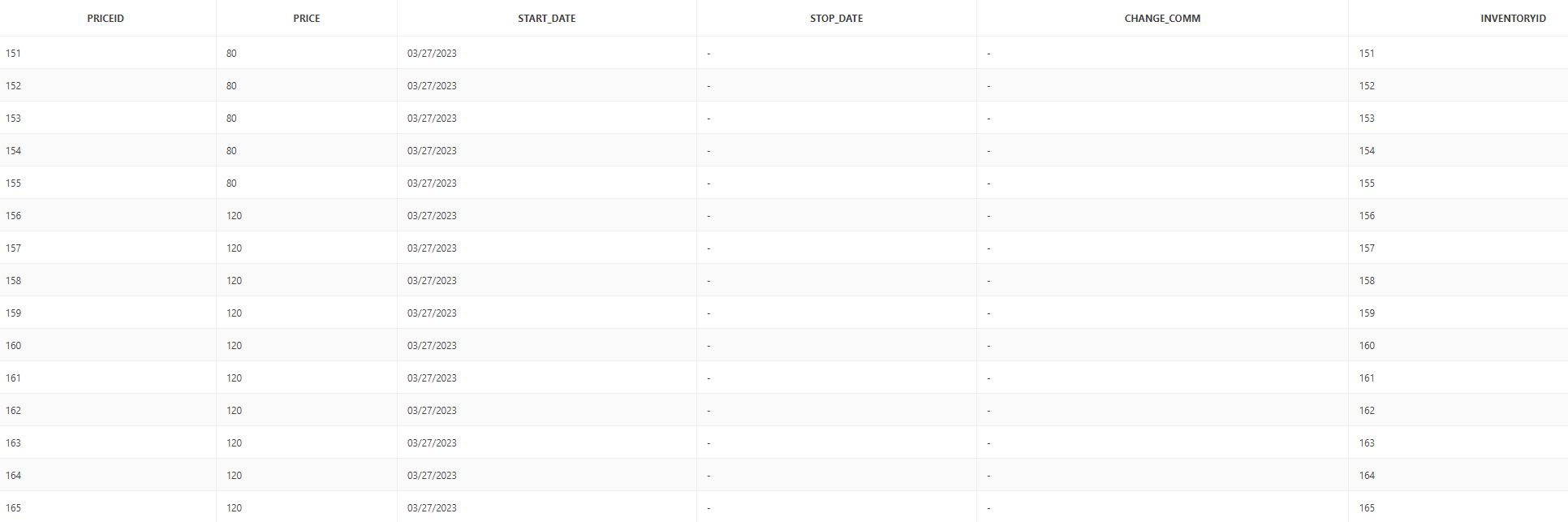


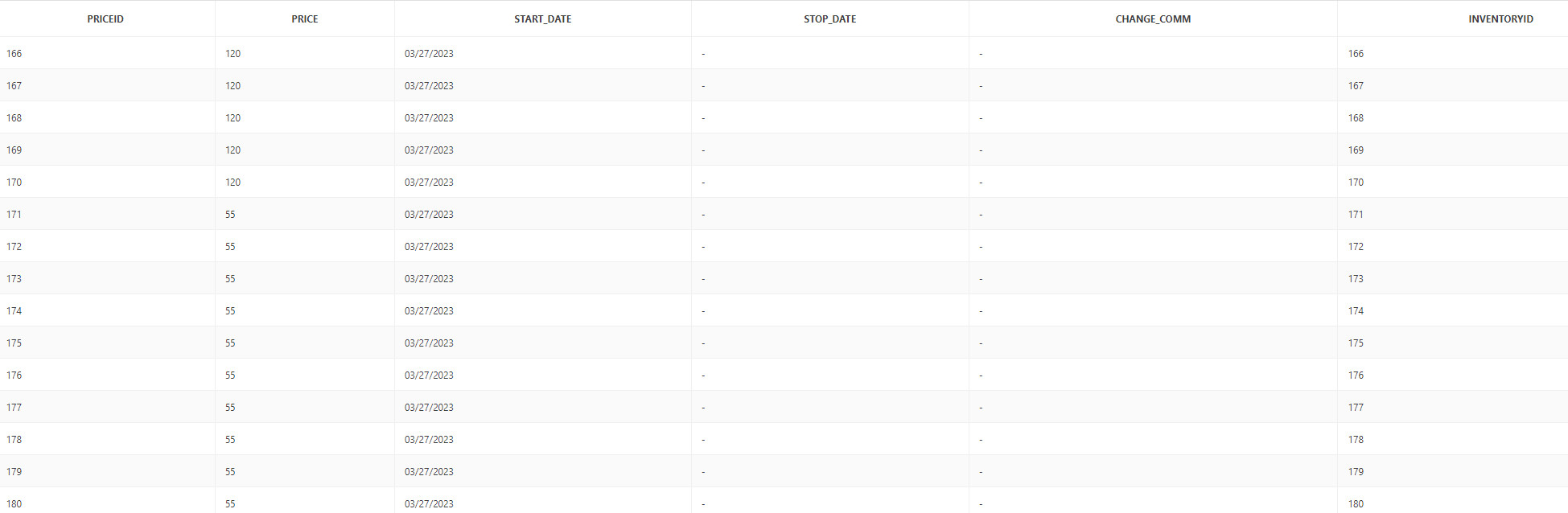


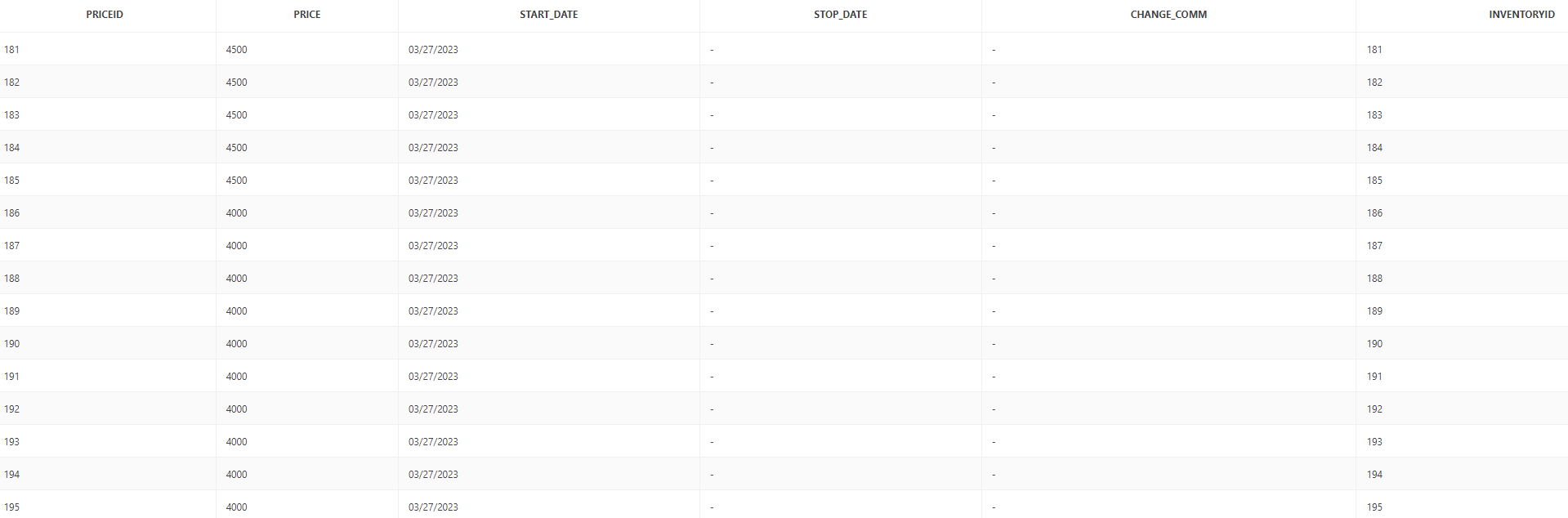


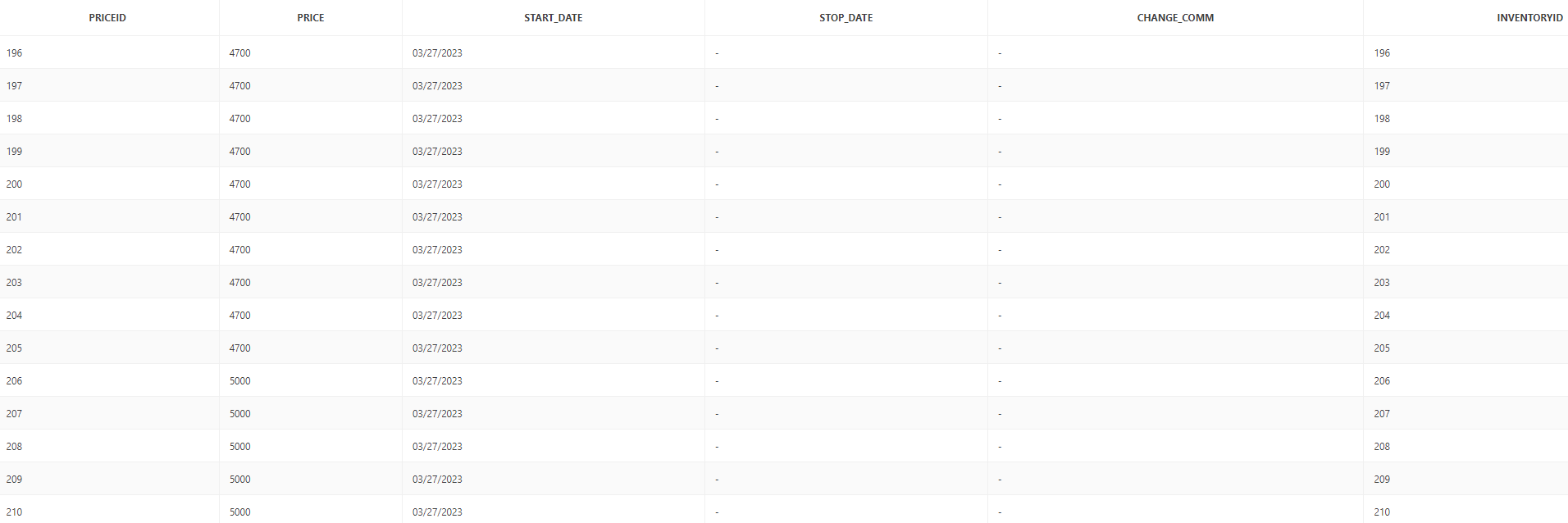








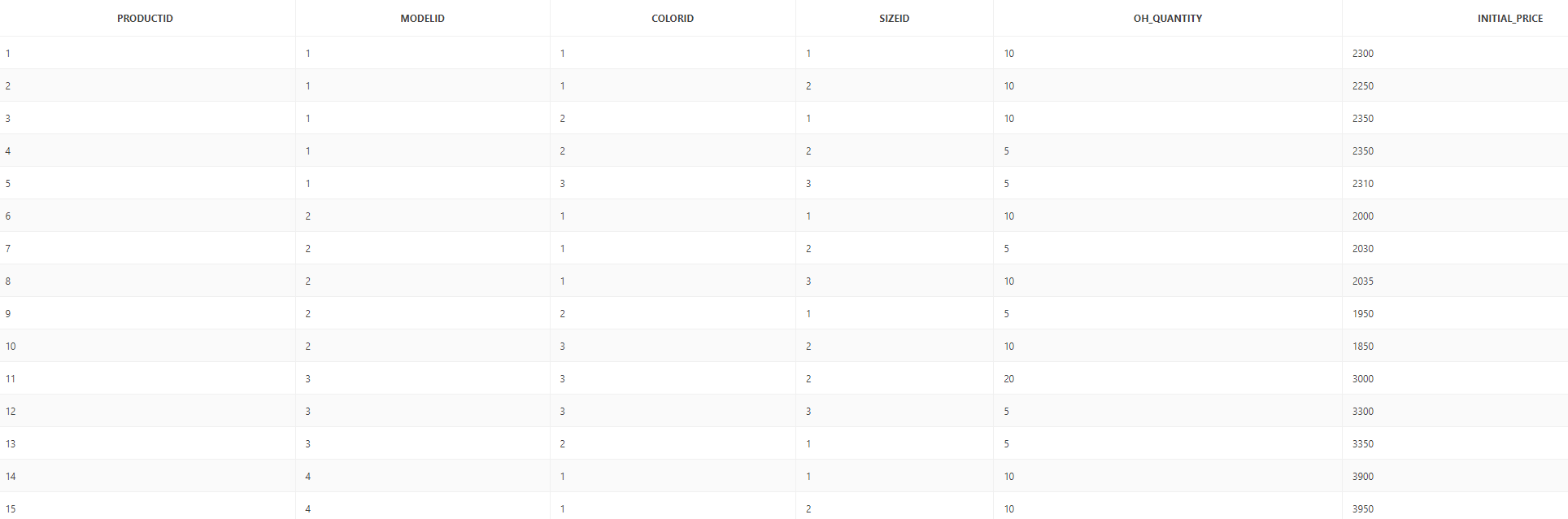


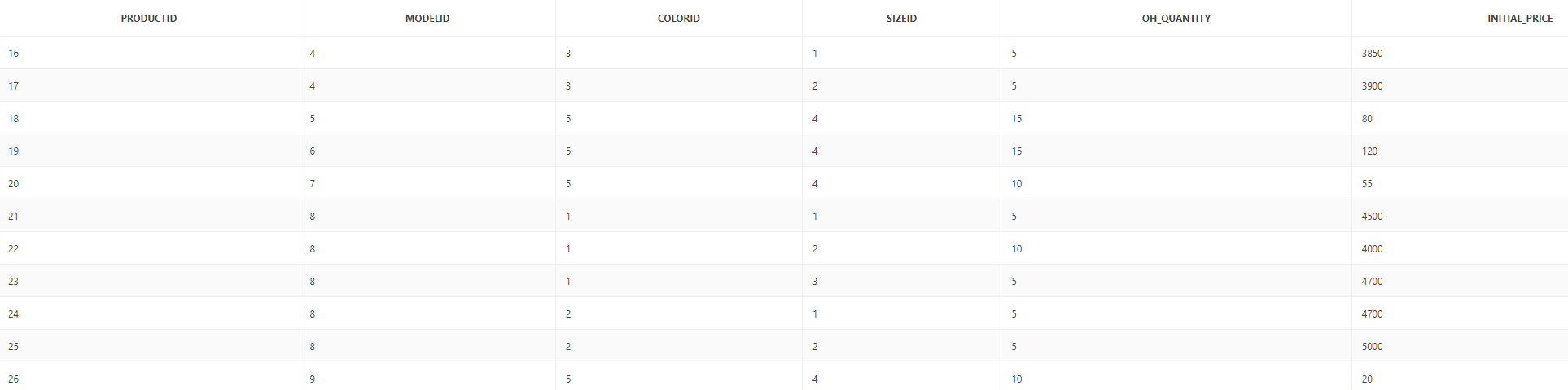




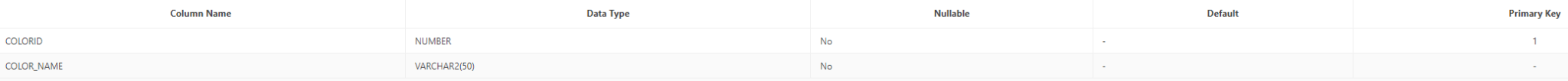
PRODUCT\_BUILT







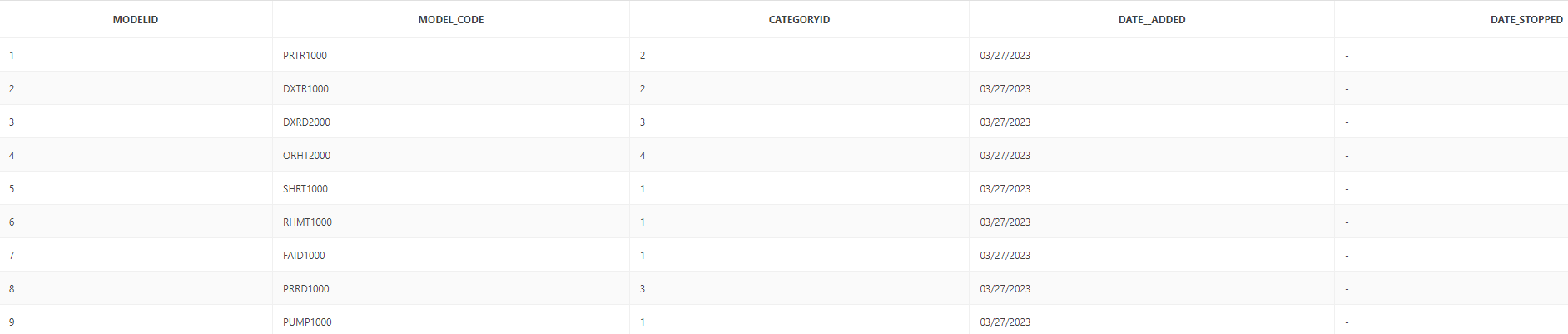
PRODUCT\_COLOR



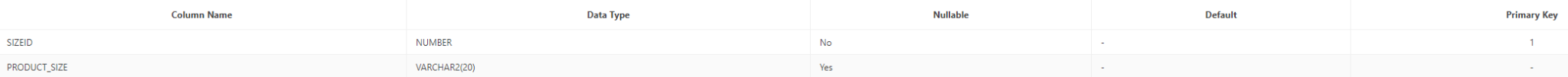


PRODUCT\_MODEL





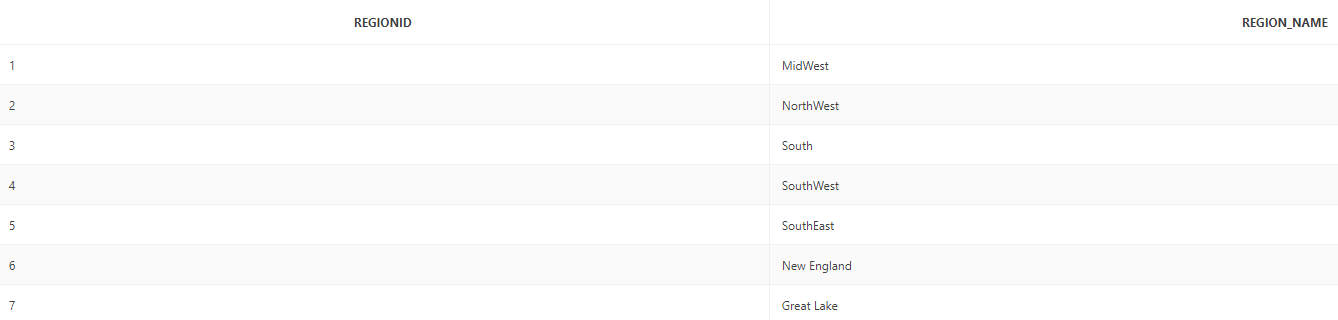
PRODUCT\_SIZE





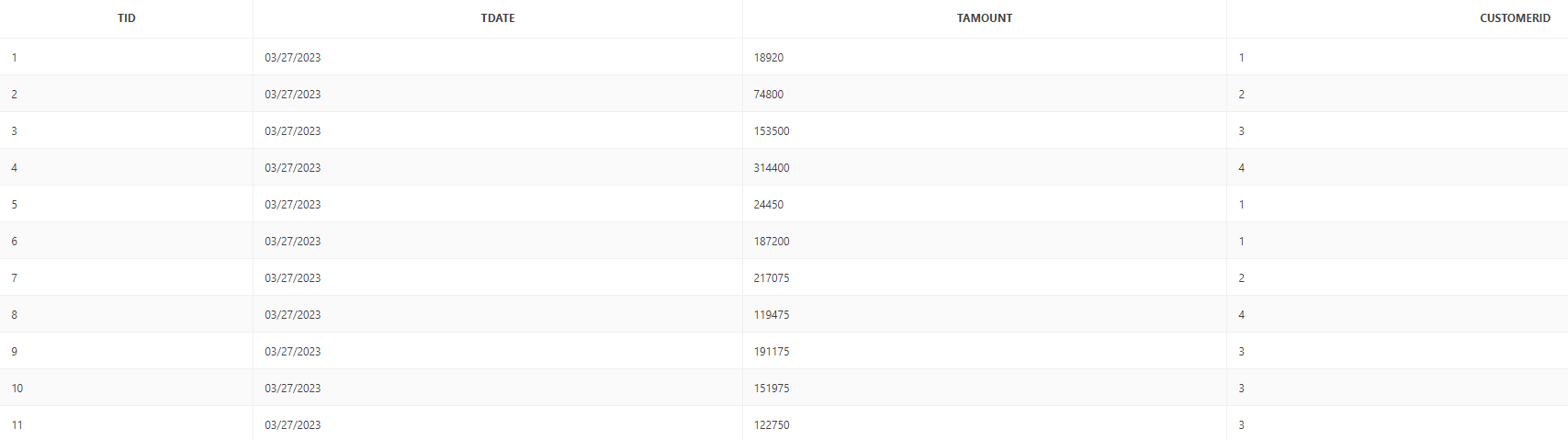
REGION





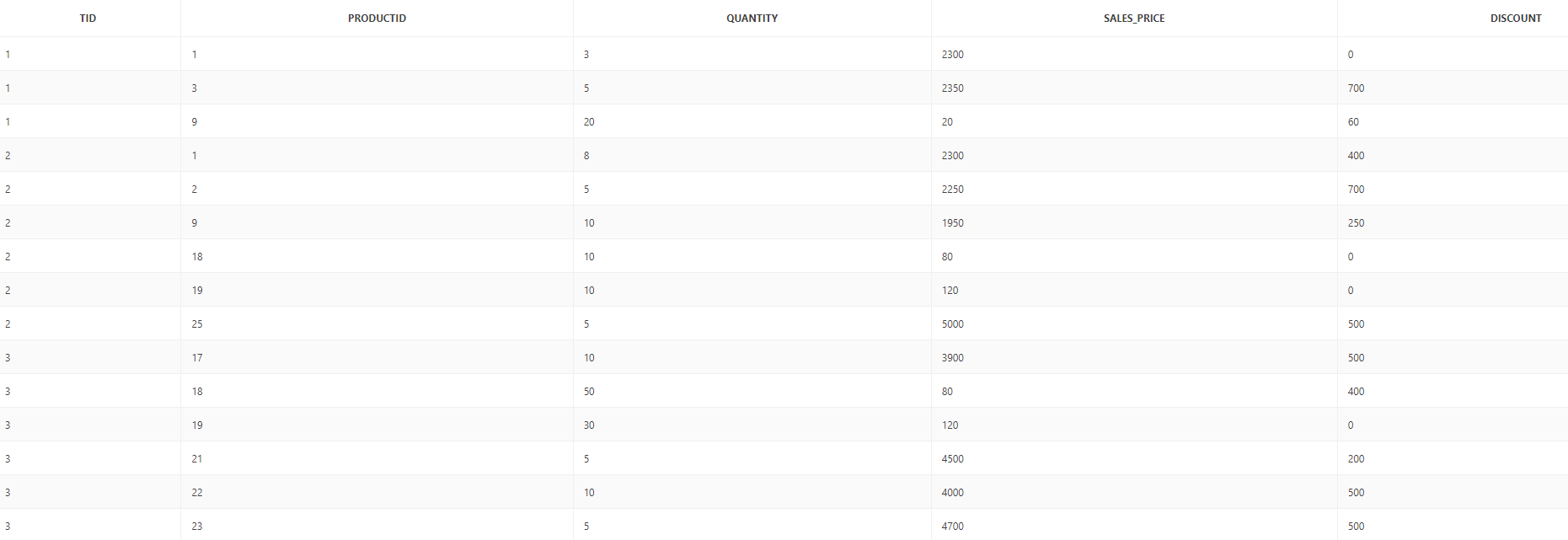
SALES\_TRANSACTION

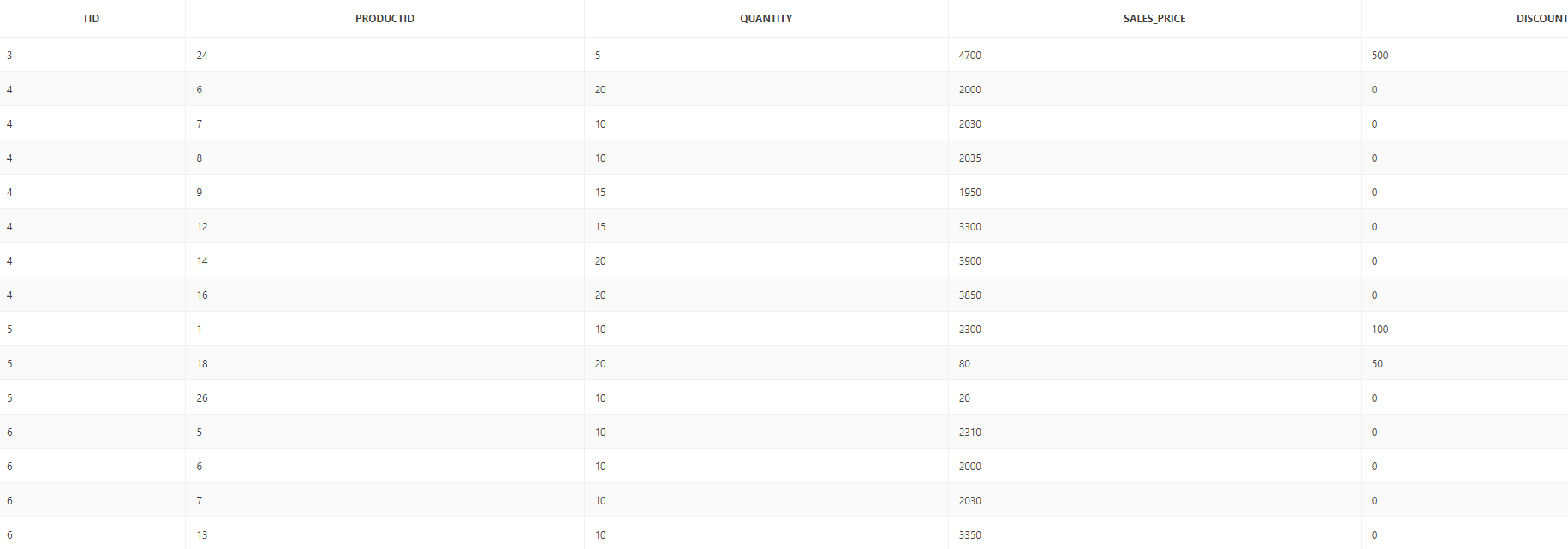


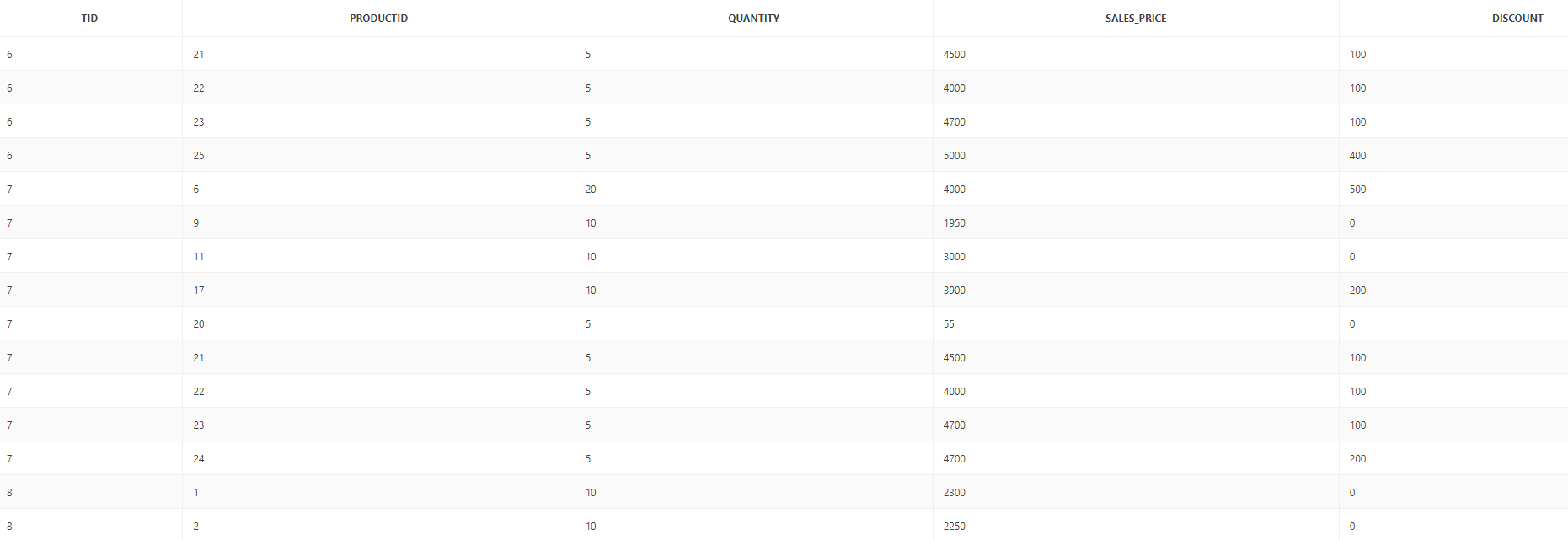


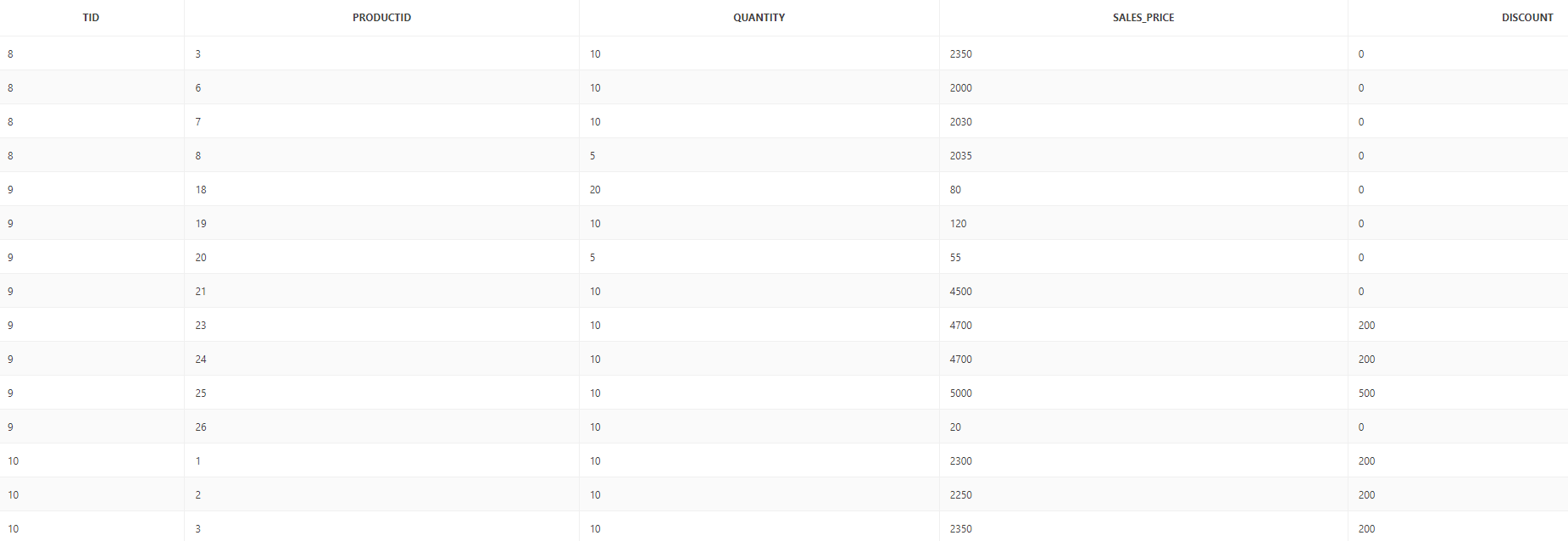
TRANSACTION\_LINE

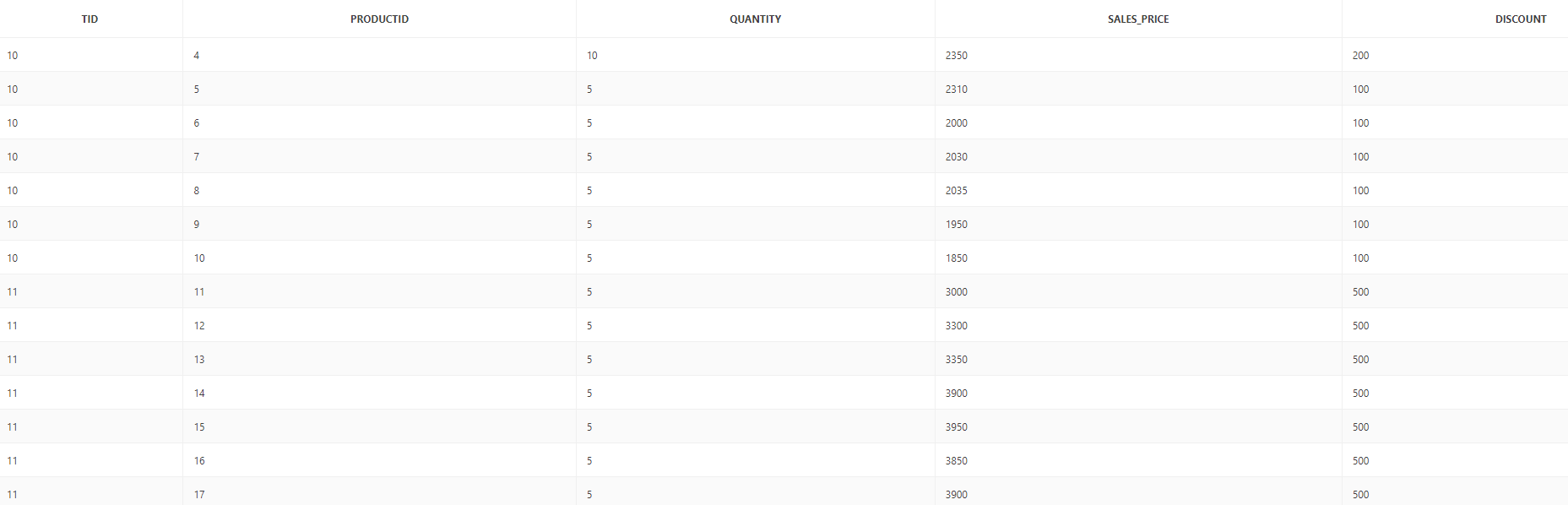




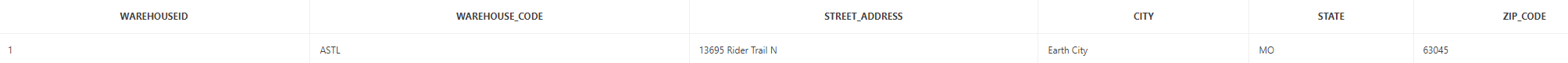








WAREHOUSE  

**Reports:**

