Super Mart Chain Database

ABSTRACT

1.1 Introduction

This Database Design Project illustrates the design process, implementation, and features of a database used to store large amounts of data for a new retail corporation with similar operations to Costco and Walmart. ABC, an upcoming national corporation, will offer both online and offline ordering options, chain retail stores nationwide, and a membership option to its customers. Like Costco, ABC receives its product inventory from manufacturers, stocks its various stores, and sells the products to its customers via in-store or online.

1.2 Overview

Like any large retail corporation, ABC will be required to manage large amounts of data, regarding its sales, customers, employees, and stores. To support ABC's business structure, not only will the database be required to collect, manage, and store data, it must also store the data correctly in order for it to be extracted and analyzed for business analytical purposes. Since ABC uses a variety of methods to sell products, the database will also need to manage all online transactions, shipment information, and payment information used in online purchases. Additionally, ABC's customers have the ability to opt for a membership option, which includes various benefits much like those at Costco. This operation will require the database to properly collect and sort data regarding the customer base, payment methods, transactions, and membership information.

1.3 Business Understanding

In order to avoid issues with data inconsistencies and discrepancies, the collection of data must be specific, complete, and consistent across all components of the business. The long-term use of a properly implemented database will further ABC's corporate reach and expand its customer base by generating direct data to syndicated data. While this database collects direct data, such as customer, store, and manufacturer information, it can also be applied to analyze trends within the data that will help ABC predict market trends, manage consumer behavior, and form strategies to compete with larger multinational retail corporations. Databases for retailers provide mutual benefit to the corporation itself and to its customers. While ABC can use the stored data to track its business activities, customers are able to view product availability, store information, and make purchases online.

ASSUMPTIONS & CONSTRAINTS

1.1 Assumptions

Every store can have 1 or 0 employees and similarly every manufacturer can have 1 or 0 products. A customer need not necessarily be a member and so does the order need not necessarily be an online order.

1.2 Constraints

Store_has_products and order_has_products are associative entities (i.e. join tables) and as such do not meet the five attribute minimum.

DESIGN DECISIONS

1.1 Key Factors Influencing Design

The entity-relationship model, as shown in the entity-relationship diagram further in this text, was used to determine the database's design. Order info joined with Customer table represents the main table of end users; Product and Manufacturer joined together represents the product catalog and their availability; Employee joined with Store_Info joined together represents the number of employees working for a store and their employment information.

1.2 Functional Design Decisions

The database will communicate with three systems regularly. End users' access to the database will be mediated by a security and authentication system, which will demand a set of legitimate and confirmed credentials to acquire said access. A level catalog management system will serve as a "middle man" between the database and end users, interpreting their requests and sending the appropriate SQL commands to the DBMS.

1.3 Database Management System Decisions

We have chosen Oracle Database 12c Enterprise Edition, Release 12.2.0.1.0 64-bit, operating on Linux, and implementing PL/SQL Release 12.2.0.1.0 as the DBMS for the initial database implementation. This DBMS was selected because of both its track record for stability and due to both PL/adherence SQL's to accepted SQL norms and the database team's familiarity with the technology. We will be able to modify the database to meet changing requirements mostly by using ALTER statements to add new fields to existing tables because the database is designed to be modular in nature and allow for quick table amendment. If the requirement change calls for more significant action than just changing the tables, the entity-relationship.

1.4 Security and Privacy Design Decisions

To help prevent tampering with player and level data, the system will employ two layers of protection. A system for login and password-based authentication will review credentials provided by the end user. Depending on the findings of the analysis, the system will decide whether to grant or refuse access to other users. After user authentication, mediator systems will moderate database commands by translating them into PL/SQL from a proprietary, non-SQL format. Using this method, end users won't be given direct access to the database and won't get useful information about its structure.

Three user types will be available on the system. The first is end users, who can never access the database directly and must constantly employ intermediaries. Although they will frequently need to use moderator systems, internal users like maintenance staff and service moderators will have more direct access to the database. And finally, complete database access will always be available to the moderator systems themselves.

STATEMENT OF WORK

1.1 Overview

This project will lay the design and creation of a relational database model used to track all the business activities of a supermarket chain like Costco, Walmart etc. The information stored includes details about stores managing products, product placements, employees, orders and will serve as a one-stop solution to all the business activities for a supermarket retailer. This database will be of help both for the customers and for the store executives. For the customers it will help track product availability, store details such as operational hours, and track an online order. For the store executives, it will help ease employee management, store management, order management and inventory management. It will thus lead to happier customers and overall success of the store.

1.2 Purpose and Objective

The database will enable the creation and maintenance of a database that'll be useful to both the consumers and the store operators. It will serve as a dynamic backend support to the supermarket website for the consumers to track the availability of a product, delivery times, store details, order history, product catalog along with discounts, while managing their membership and payment details (only for members), all at one place. On the consumer end, the website will be designed on JavaScript and a cloud based database server. For the store operators, the database will be embedded in the inventory and store management software installed on-site at the store with real-time updates to track stock inventory, orders, shipments, and employees. The inventory/store management system will be designed using JAVA and Php/MyAdmin.

1.3 Project Scope

The database will act as a way for better shopping experience for the consumers and an error-free accurate management system for the store executives. However, the scope of this project is limited to the design of the database implementing appropriate relationships between different entities of the database. The In-scope work will include documentation of project requirements, designing an ERD model-that accurately represents entities, their attributes and relationships- using crow's foot notation, writing Data Definition Language (DDL) for defining different aspects of the database including but not limited to entities, and providing sample Data Manipulation Language (DML) and Structured Query Language (SQL) for presenting probable use of this database. The tasks including the design of the consumer websites and inventory/story management software- using JAVASCRIPT and JAVA/Php- and connecting it to the server backend for structured querying is outside the scope of this project.

1. **In-Scope**
    - Project documentation
    - Designing Entity-Relationship diagram in an RDBMS using Crow's foot Notation
    - Implementing DDL
    - Providing sample DML
    - Providing sample SQL
    - Writing a comprehensive project report

2. **Out-of-Scope**

   - Developing the inventory/store management system
   - Developing the consumer website
   - Providing admin access

1.4 Database Goals, Expectations and Deliverables

Proper implementation of this project will yield a database that shall hold all the metadata, as well as the entities, their primary keys, attributes with proper constraints on datatypes and their relationships. Surrogate keys and entity super-type/sub-type relations will be implemented wherever required. Normalization techniques will be practiced to reduce data redundancy. Deliverables included a detailed report, an Entity-Relationship diagram using Crow's Foot notation, Data Definition Language script, Example Data Manipulation Script and SQL script.

1.5 Database Benefits

Benefits of the database can be categorized into consumer benefits for a smooth shopping experience, and ease of management benefits for the store operators. For the consumers, the benefits include but are not limited to tracking store details, a product's price, availability, costs, and offers, and tracking individual orders for shipping info, payment status, and order status, all integrated on the consumer website for real-time dynamic updates. For store operators, the database will provide easy access to track current inventory, demand, manufacturer records, current orders, past orders, and product placement, while providing a medium of proper employee management and tracking.

1.6 Project Hardware and Software Tools

Diagram Tool

Draw.io, running on Macbook Monterey

Office Productivity Tools

Microsoft Office 365, running on Macbook Monterey

Database

Oracle Database 11g, using Virtual Desktop Access (VDA)

Hardware and Software

M1 Pro, 10 core CPU, 14 Core GPU, Apple Silicon, AMD x86 running macOS

Client Access Method

In a SQL client side browser, on a consumer-class computer running on any OS such as Windows, Linux or macOS, client software will send HTTP requests to the client side server, which translates them to SQL DML requests. The results are then processed by the server and are transmitted to the client using HTTP.

1.7 SQL Usage and Style

General

- Use underscores instead of camel cases.
- Avoid using Hungarian notation
- Avoiding naming redundancy between entity names and attributes.
- Add unique constraints to attributes where unique constraints apply.

- Use surrogate primary keys to avoid business relevant information influencing primary keys.
- Maintain NOT NULL constraints wherever a NULL value can lead to corrupt data.
- Avoiding multiple pieces of information in a single field.
- Use thoughtful maximum sizes for columns.
- Don't create large tables just to avoid joins for performance and instead use entity super-type subtype relations wherever appropriate.
- Use two separate text fields for names.
- Ensure the limit for E-mail is 320 to handle the maximum allowed length.
- Store different segments of address in separate attributes.
- Store time and date formation in ISO-8601 format (YYYY-MM-DD HH:MM:SS.SSSSS).
- Use C-style comments with opening /* and closing /* digraphs whenever possible; otherwise, precede comments with -- and finish them with a new line.
- Avoid applying object-oriented design principles to SQL or database structures.

Naming Conventions

- Avoid Spaces and quotes in Object Names.
- Maintain single forms in object Names.
- Maintain lower case letters for object names.
- Avoiding reserved/keywords for object names.
- Naming the bridge table for what it represents

- Specifically maintain constraint names.
- Use consistent and descriptive identifiers and names.

Query Syntax and Formatting

- Make use of Aliases as the size of the query grows.
- Use a consistent case for keywords for all queries.
- Use a new line for every new keyword component of a query.
- Columns inside a SELECT clause should be in their own lines.
- Make the keywords left-aligned.
- Don't indent columns in a JOIN
- Use multiple lines for joins on more than one attribute.
- Indent subqueries.
- Avoid using BETWEEN for dates.
- Use IN instead of multiple OR.
- Don't use direct functions inside a WHERE clause.
- Use NOT EXISTS instead of NOT IN
- Use UNION ALL instead of UNION if possible.
- Use HAVING only with aggregate functions
- Always specify column names in INSERT clause.
- Avoid using wildcards where possible.
- Avoid database management system-specific keywords when an ANSI SQL equivalent already exists.

- Spaces should be used to line up code so that the root keywords all end on the same character boundary. This will improve the ability to scan the code quickly.
- Use the CASE expression to interpret values before leaving the database.

Create Syntax

- Avoid using FLOAT data types wherever possible.
- Avoid vendor-specific data types whenever possible, as they are not portable.
- Keys should be chosen from those columns whose data types are less likely to change in the future.
- All entities must have at least one key.
- Apply normalization for all cases. Use compound keys only where necessary.
- Use LIKE and SIMILAR TO constraints to ensure the integrity of strings whose format is known.
- If the range of a numerical value is known, CHECK() should be used to avoid incorrect values or the silent truncation of data.
- Avoid using separate columns for units.

1.8 Project Management Methodology

The initial database design may be executed in a straightforward manner similar to the interpretations of the waterfall model. The first database requirements defined at the start of the parent project should be satisfied by this early implementation. On finishing that implementation, the database team should transition to a project management methodology that emphasizes quick iterations; ideally this methodology should be the same one used by the software development team, to help foster communication and cooperation between the two teams. After that, the database team should continue to iteratively alter the database architecture in response to modifications made to the software project’s design and input from the software development team.

REQUIREMENTS DEFINITION DOCUMENT

1.1 Business Rules

- A manufacturer can provide many products, but a product is provided by only one manufacturer.
- A customer can place many orders, but each order can only be linked to one customer.
- A member may use one or many payment cards but each payment card can only be used by one member.
- An order can be fulfilled by one or many shipments, but each shipment can only fulfill one unique order placed.
- One order can consist of one or many products. If an order has multiple products, there will be several rows in ORDER_HAS_PRODUCTS for that order.
- One product can be in one or many orders. If one product is in multiple orders, there will be several rows in ORDER_HAS_PRODUCT for that product.
- One store can employ many employees, but one employee can only be employed by one or zero stores.
- A store can sell one or many products. If a store sells multiple products, there will be multiple occurrences of that store in STORE_HAS_PRODUCTS

- A product can be in one or many stores. For every instance of PRODUCT and STORE, there will be an occurrence in STORE_HAS_PRODUCTS

1.2 Entity and Attribute Description

Entity Name: **CUSTOMER**

Entity Description: Supertype to MEMBER

Main Attributes of CUSTOMER:

CUSTOMER_ID (Primary Key): A unique identifier for each customer

FIRST_NAME: Customer’s first name

MIDDLE: Customer’s middle name

LAST_NAME: Customer’s last name

DOB: Customer’s date of birth

PHONE: Numeric phone number value for each customer.

STREET: Alphanumeric street address for each customer.

CITY: City in which the customer resides

ZIPCODE: Numeric zip code value for each customer.

EMAIL: Alphanumeric email address for each customer.

STATE_LOC: State in which the customer resides

Entity Name: **PRODUCT**

Entity Description: A product is provided by a manufacturer and purchased by a customer or member.

Main Attributes of PRODUCT:

PRODUCT_ID (Primary Key): A unique identifier for each product

PRODUCT_NAME: Display name of the product.

ManufacturerID (Foreign Key): The unique ID for the manufacturer that provided the product.

PRICE: Numeric price of a product

INVENTORY: Value of available quantity for each product.

TAXABLE: Percentage amount of sales tax associated with a product

DISCOUNT:

Entity Name: **MEMBER**

Entity Description: If the customer holds a membership or not. Related to Payment info. A SUBTYPE of CUSTOMER.

Main Attributes of MEMBER:

MEMBERSHIP_NO (Primary Key): A unique identifier for each member

MEMBERSHIP_DATE: Date when a member activates a membership

Entity Name: **ALL_ORDER**

Entity Description: Supertype to ONLINE_ORDER

Main Attributes of ORDER:

ORDER_ID (Primary Key): A unique identifier for each processed order

CUSTOMER_ID (Foreign Key): A unique identifier for each customer, used to link the order with the customer that made the order.

ORDER_TYPE: Shows whether the order was placed ONLINE or IN-STORE.

ORDER_DATE: The date when the order was placed.

NO_OF_ITEMS: Total quantity of items placed in an order.

TOTAL_AMT: The total value in dollars of products placed in an order.

Entity Name: **MANUFACTURER**

Entity Description: Manufacturers for a particular product.

Main Attributes of MANUFACTURER:

<u>MANUFACTURER_ID</u> (Primary Key): A unique identifier for each manufacturer.

MANUFACTURER_NAME: Name of the manufacturer company.

STREET: The alphanumerical street address where the manufacturer is located.

CITY: The city in which the manufacturer is located.

COUNTRY: The country in which the manufacturer is located. (Country in which a product is shipping from)

EMAIL: Alphanumeric email address for each manufacturer.

PHONE: A manufacturer's phone number.

Entity Name: **STORE_INFO**

Entity Description: A STORE employs an EMPLOYEE and sells PRODUCT.

Main Attributes of STORE_INFO:

<u>STORE_ID:</u> (Primary Key) A unique identifier for a store location. Provides saved store information.

STORE_LOC: Abbreviated display name using a store's location that identifies a store.

STREET: Alphanumeric street address of the store

CITY: City in which the store is located

STATE_NAME: State in which the store is located

AREA_IN_SQFT: Numeric value displaying the area of the store's area in square feet.

LIQUOR_SALES: Amount in dollars collected from hard liquor sales at each store.

Entity Name: **EMPLOYEE**

Entity Description:

Main Attributes of EMPLOYEE:

EMPLOYEE_ID(Primary Key): A unique identifier for each employee, used as unique login credentials as username

FIRST_NAME: The employee's first name

MIDDLE: The employee's middle name

LAST_NAME: The employee's last name

DATE_EMPLOYED: Date when the employee began employment at the store.

SSN: The employee's unique SSN, used for employment purposes, values are encrypted in data

DOB: The employee's date of birth

EMAIL: Alphanumeric email address for each employee

PHONE: The employee's phone number

STREET: An employee's street address (Alphanumeric)

CITY: The city in which an employee resides

ZIPCODE: The zip code in which an employee resides

STATE_LOC: The state in which an employee resides

Entity Name: **PAYMENT_INFO**

Entity Description: Saved payment information associated with each member

Main Attributes of PAYMENT_INFO:

PAYMENT_ID (Primary Key): A unique identifier associated with the saved payment information for each member (includes CardNumber, CVV, Export, CardType)

CARD_NUMBER: A unique numeric credit/debit card number

CVV: A unique security code for the associated payment card.

EXPIRY: Expiration date value for the associated payment card.

CARD_TYPE: The payment processor for the associated payment card. (ie: Visa, MasterCard, American Express, or Discover)

MEMBERSHIP_NO (Foreign Key): A unique identifier for each member, used to link payment information to members.

Entity Name: **ONLINE_ORDER**

Entity Description: A SUBTYPE of ALL_ORDER

Main Attributes of ONLINE_ORDER:

REFERENCE_ID (Primary Key): A unique identifier for each order made ONLINE.

Entity Name: **SHIPMENT**

Entity Description:

Main Attributes of SHIPMENT:

SHIPMENT_ID (Primary Key): A unique identifier for the shipment information of an order.

SHIPMENT_STATUS: Provides the status of the shipment; Processing, Shipped, Delivered, Failed

DATE_SHIPPED: The date in which the shipment was successfully shipped.

TARGET_DELIVERY: The predicted date in which the shipment will be successfully delivered to customer/member.

Entity Name: **STORE_HAS_PRODUCTS**

Entity Description: A bridge table linking STORE_INFO to PRODUCT.

Main Attributes of STORE_HAS_PRODUCTS:

STORE_ID (Foreign Key):

PRODUCT_ID (Foreign Key): A unique identifier for a product.

Entity Name: **ORDER_HAS_PRODUCTS**

Entity Description: A bridge table linking PRODUCT to ALL_ORDER.

Main Attributes of ORDER_HAS_PRODUCTS:

PRODUCT_ID (Foreign Key): A unique identifier for each product.

ORDER_ID (Foreign Key)

1.3 Relationship and Cardinality Description

Relationship: Provides between MANUFACTURER and PRODUCT

Cardinality: 1:M between MANUFACTURER and PRODUCT

Business Rules: A manufacturer can provide many products, but a product is provided by only one manufacturer.

Relationship: Places between CUSTOMER and ALL_ORDER

Cardinality: 1:M between CUSTOMER and ALL_ORDER

Business Rules: A customer can place many orders, but each order can only be linked to one customer.

Relationship: Uses between MEMBER and PAYMENT_INFO

Cardinality: 1:M between MEMBER and PAYMENT_INFO

Business Rules: A member may use one or many payment cards but each payment card can only be used by one member.

Relationship: Fulfills between ONLINE_ORDER and SHIPMENT

Cardinality: 1:M between ONLINE_ORDER and SHIPMENT

Business Rules: An order can be fulfilled by one or many shipments, but each shipment can only fulfill one unique order placed.

Relationship: Consists of between ALL_ORDER and ORDER_HAS_PRODUCTS

Cardinality: 1:M between ALL_ORDER and ORDER_HAS_PRODUCTS

Business Rules: One order can consist of one or many products. If an order has multiple products, there will be several rows in ORDER_HAS_PRODUCTS for that order.

Relationship: In between PRODUCT and ORDER_HAS_PRODUCTS

Cardinality: 1: M between PRODUCT and ORDER_HAS_PRODUCTS

Business Rules: One product can be in one or many orders. If one product is in multiple orders, there will be several rows in ORDER_HAS_PRODUCT for that product.

Relationship: Employs between STORE_INFO and EMPLOYEE

Cardinality: 1:M between STORE_INFO and EMPLOYEE

Business Rules: One store can employ many employees, but one employee can only be employed by one or zero stores.

Relationship: Sells between STORE_INFO and STORE_HAS_PRODUCTS

Cardinality: 1:M between STORE_INFO and STORE_HAS_PRODUCTS

Business Rules: A store can sell one or many products. If a store sells multiple products, there will be multiple occurrences of that store in STORE_HAS_PRODUCTS

Relationship: Has between PRODUCT and STORE_HAS_PRODUCTS

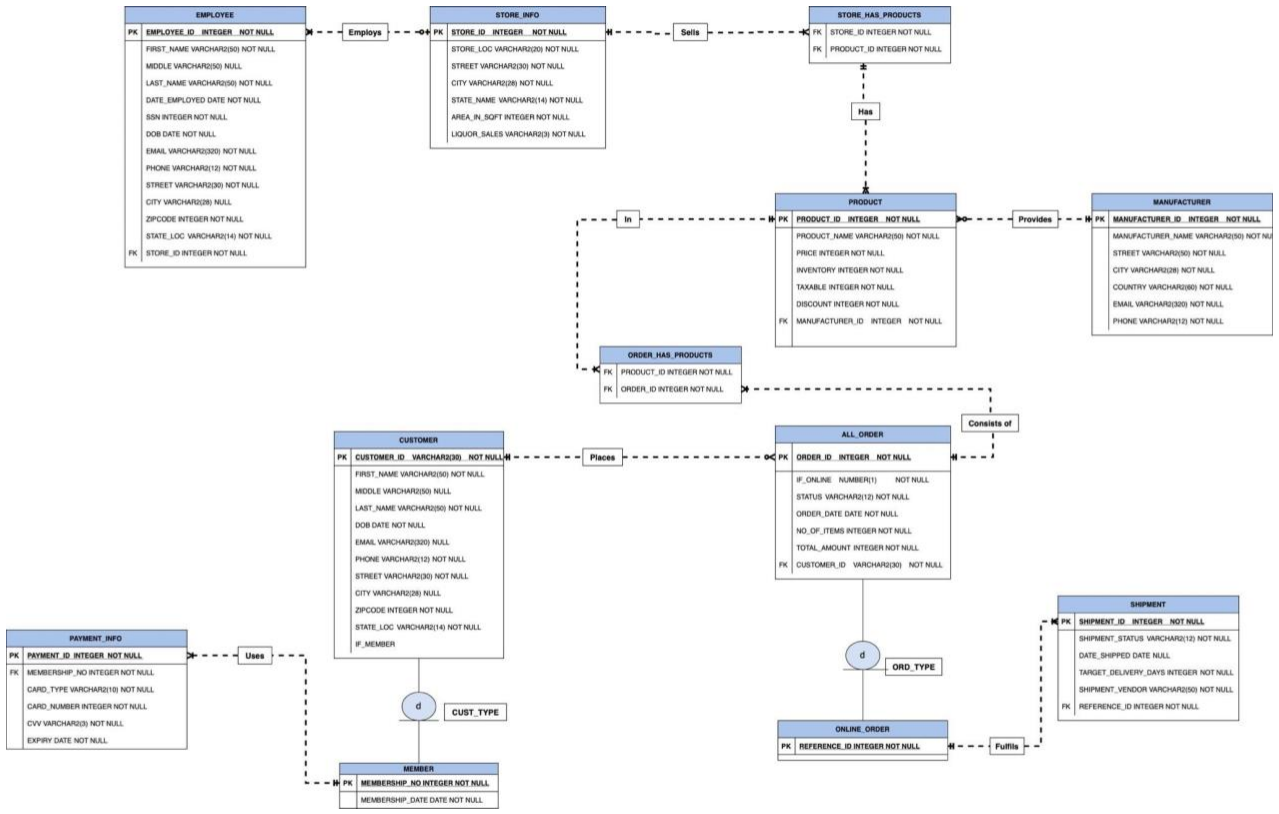
Cardinality: 1:M between STORE_HAS_PRODUCTS and PRODUCT

Business Rules: A product can be in one or many stores. For every instance of PRODUCT and STORE, there will be an occurrence in STORE_HAS_PRODUCTS

DETAILED DATABASE DESIGN

1.1 Entity Relationship Diagram (ERD):

(Implement a supertype-subtype relation between customers-members and orders-online)


EMPLOYEE
PK EMPLOYEE_ID INTEGER NOT NULL
FIRST_NAME VARCHAR2(50) NOT NULL
MIDDLE VARCHAR2(50) NULL
LAST_NAME VARCHAR2(50) NOT NULL
DATE_EMPLOYED DATE NOT NULL
SSN INTEGER NOT NULL
DOB DATE NOT NULL
EMAIL VARCHAR2(320) NOT NULL
PHONE VARCHAR2(12) NOT NULL
STREET VARCHAR2(30) NOT NULL
CITY VARCHAR2(28) NULL
ZIPCODE INTEGER NOT NULL
STATE_LOC VARCHAR2(14) NOT NULL
FK STORE_ID INTEGER NOT NULL
Employs
STORE_INFO
PK STORE_ID INTEGER NOT NULL
STORE_LOC VARCHAR2(20) NOT NULL
STREET VARCHAR2(30) NOT NULL
CITY VARCHAR2(28) NOT NULL
STATE_NAME VARCHAR2(14) NOT NULL
AREA_IN_SQFT INTEGER NOT NULL
LIQUOR_SALES VARCHAR2(3) NOT NULL
Sells
STORE_HAS_PRODUCTS
FK STORE_ID INTEGER NOT NULL
FK PRODUCT_ID INTEGER NOT NULL
Has
PRODUCT
PK PRODUCT_ID INTEGER NOT NULL
PRODUCT_NAME VARCHAR2(50) NOT NULL
PRICE INTEGER NOT NULL
INVENTORY INTEGER NOT NULL
TAXABLE INTEGER NOT NULL
DISCOUNT INTEGER NOT NULL
FK MANUFACTURER_ID INTEGER NOT NULL
Provides
MANUFACTURER
PK MANUFACTURER_ID INTEGER NOT NULL
MANUFACTURER_NAME VARCHAR2(50) NOT NU
STREET VARCHAR2(50) NOT NULL
CITY VARCHAR2(28) NOT NULL
COUNTRY VARCHAR2(60) NOT NULL
EMAIL VARCHAR2(320) NOT NULL
PHONE VARCHAR2(12) NOT NULL
In
ORDER_HAS_PRODUCTS
FK PRODUCT_ID INTEGER NOT NULL
FK ORDER_ID INTEGER NOT NULL
Consists of
CUSTOMER
PK CUSTOMER_ID VARCHAR2(30) NOT NULL
FIRST_NAME VARCHAR2(50) NOT NULL
MIDDLE VARCHAR2(50) NULL
LAST_NAME VARCHAR2(50) NOT NULL
DOB DATE NOT NULL
EMAIL VARCHAR2(320) NULL
PHONE VARCHAR2(12) NOT NULL
STREET VARCHAR2(30) NOT NULL
CITY VARCHAR2(28) NULL
ZIPCODE INTEGER NOT NULL
STATE_LOC VARCHAR2(14) NOT NULL
IF_MEMBER
Places
ALL_ORDER
PK ORDER_ID INTEGER NOT NULL
IF_ONLINE NUMBER(1) NOT NULL
STATUS VARCHAR2(12) NOT NULL
ORDER_DATE DATE NOT NULL
NO_OF_ITEMS INTEGER NOT NULL
TOTAL_AMOUNT INTEGER NOT NULL
FK CUSTOMER_ID VARCHAR2(30) NOT NULL
PAYMENT_INFO
PK PAYMENT_ID INTEGER NOT NULL
FK MEMBERSHIP_NO INTEGER NOT NULL
CARD_TYPE VARCHAR2(10) NOT NULL
CARD_NUMBER INTEGER NOT NULL
CVV VARCHAR2(3) NOT NULL
EXPIRY DATE NOT NULL
Uses
d
CUST_TYPE
MEMBER
PK MEMBERSHIP_NO INTEGER NOT NULL
MEMBERSHIP_DATE DATE NOT NULL
d
ORD_TYPE
ONLINE_ORDER
PK REFERENCE_ID INTEGER NOT NULL
Fulfils
SHIPMENT
PK SHIPMENT_ID INTEGER NOT NULL
SHIPMENT_STATUS VARCHAR2(12) NOT NULL
DATE_SHIPPED DATE NULL
TARGET_DELIVERY_DAYS INTEGER NOT NULL
SHIPMENT_VENDOR VARCHAR2(50) NOT NULL
FK REFERENCE_ID INTEGER NOT NULL

1.2 DDL SOURCE CODE

```
/*
DDL QUERIES
*/
-- SEQUENCE
DROP SEQUENCE SEQ_STORE_ID;
DROP SEQUENCE SEQ_PRODUCT_ID;
DROP SEQUENCE SEQ_EMPLOYEE_ID;
DROP SEQUENCE SEQ_MANUFACTURER_ID;
DROP SEQUENCE SEQ_MEMBERSHIP_NO;
DROP SEQUENCE SEQ_ORDER_ID;
DROP SEQUENCE SEQ_REFERENCE_ID;
DROP SEQUENCE SEQ_PAYMENT_ID;
DROP SEQUENCE SEQ_SHIPMENT_ID;

-- INDICES
DROP INDEX IDX_EMPLOYEE_ZIPCODE;

DROP INDEX IDX_MANUFACTURER_COUNTRY;

DROP INDEX IDX_PRODUCT_MANUFACTURER;
DROP INDEX IDX_PRODUCT_PRICE;
DROP INDEX IDX_PRODUCT_INVENTORY;
DROP INDEX IDX_PRODUCT_TAXABLE;

DROP INDEX IDX_CUSTOMER_PHONE;
DROP INDEX IDX_CUSTOMER_ZIPCODE;
DROP INDEX IDX_CUSTOMER_DOB;
DROP INDEX IDX_CUSTOMER_STATE;

DROP INDEX IDX_PAYMENT_CARD_TYPE;

DROP INDEX IDX_ALL_ORDER_USER_ID;
DROP INDEX IDX_ALL_ORDER_STATUS;
DROP INDEX IDX_ALL_ORDER_DATE;
DROP INDEX IDX_ALL_ORDER_TOTAL_AMOUNT;

DROP INDEX IDX_SHIPMENT_REFERENCE_ID;
DROP INDEX IDX_SHIPMENT_STATUS;
DROP INDEX IDX_SHIPMENT_TARGET_DELIVERY;
DROP INDEX IDX_SHIPMENT_VENDOR;
```

```
--TABLES

DROP TABLE ORDER_HAS_PRODUCTS;
DROP TABLE STORE_SELL_PRODUCTS;
DROP TABLE PRODUCT;
DROP TABLE MANUFACTURER;
DROP TABLE EMPLOYEE;
DROP TABLE PAYMENT_INFO;
DROP TABLE SHIPMENT;
DROP TABLE ALL_ORDER;
DROP TABLE CUSTOMER;
DROP TABLE STORE_INFO;


CREATE TABLE STORE_INFO(
   STORE_ID          INTEGER           NOT NULL,
   STORE_LOC         VARCHAR2(20)      NOT NULL UNIQUE,
   STREET            VARCHAR2(30)      NOT NULL,
   CITY              VARCHAR2(28)      NOT NULL,
   STATE_NAME        VARCHAR2(14)      NOT NULL,
   AREA_IN_SQFT      INTEGER           NOT NULL,
   LIQUOR_SALES      NUMBER(1)         NOT NULL CHECK  (LIQUOR_SALES IN (0,1)),

   CONSTRAINT        PK_STORE_INFO   PRIMARY KEY (STORE_ID)
);

CREATE TABLE EMPLOYEE(
   EMPLOYEE_ID       INTEGER           NOT NULL,
   STORE_ID          INTEGER           NOT NULL,
   FIRST_NAME        VARCHAR2(50)      NOT NULL,
   MIDDLE            VARCHAR2(50)      ,
   LAST_NAME         VARCHAR2(50)      NOT NULL,
   DATE_EMPLOYED     DATE              NOT NULL,
   SSN              INTEGER           NOT NULL UNIQUE,
   DOB              DATE              NOT NULL,
   EMAIL            VARCHAR2(320)     NOT NULL UNIQUE,
   PHONE            VARCHAR2(12)      NOT NULL,
   STREET           VARCHAR2(30)      NOT NULL,
   CITY             VARCHAR2(28)      ,
   ZIPCODE          INTEGER           NOT NULL,
   STATE_LOC        VARCHAR2(14)      NOT NULL,


   CONSTRAINT PK_EMPLOYEE_     PRIMARY KEY (EMPLOYEE_ID),
   CONSTRAINT FK_STORE_EMPLOYED FOREIGN KEY (STORE_ID) REFERENCES
STORE_INFO
);

CREATE TABLE MANUFACTURER(
   MANUFACTURER_ID      INTEGER           NOT NULL,
   MANUFACTURER_NAME    VARCHAR2(50)      NOT NULL,
   STREET               VARCHAR2(50)      NOT NULL,
   CITY                 VARCHAR2(28)      NOT NULL,
   COUNTRY              VARCHAR2(60)      NOT NULL,
   EMAIL                VARCHAR2(320)     NOT NULL,
   PHONE                VARCHAR2(12)      NOT NULL,

   CONSTRAINT      PK_MANUFACTURER    PRIMARY KEY (MANUFACTURER_ID)
);

CREATE TABLE PRODUCT(
   PRODUCT_ID         INTEGER           NOT NULL,
   MANUFACTURER_ID    INTEGER           NOT NULL,
   PRODUCT_NAME       VARCHAR2(50)      NOT NULL,
   PRICE              INTEGER           NOT NULL,
   INVENTORY          INTEGER           NOT NULL,
   TAXABLE            INTEGER           NOT NULL,
   DISCOUNT            INTEGER             NOT NULL,

   CONSTRAINT  PK_PRODUCTS_      PRIMARY KEY (PRODUCT_ID),
   CONSTRAINT  FK_PRODUCT_MANUFACTURER_ID  FOREIGN KEY (MANUFACTURER_ID)
REFERENCES MANUFACTURER
);

CREATE TABLE CUSTOMER(
   USER_ID             VARCHAR2(30)        NOT NULL,
   IF_MEMBER           NUMBER(1)           NOT NULL CHECK  (IF_MEMBER IN (0,1)),
   FIRST_NAME          VARCHAR2(50)        NOT NULL,
   MIDDLE              VARCHAR2(50)        ,
   LAST_NAME           VARCHAR2(50)        NOT NULL,
   DOB                 DATE                NOT NULL,
   EMAIL               VARCHAR2(320)       ,
   PHONE               VARCHAR2(12)     NOT    NULL,
   STREET              VARCHAR2(30)     NOT    NULL,
   CITY                VARCHAR2(28)      ,
   ZIPCODE             INTEGER             NOT NULL,
   STATE_LOC           VARCHAR2(14)        NOT NULL,
   MEMBERSHIP_NO       VARCHAR2(10)UNIQUE,

   CONSTRAINT          PK_CUSTOMERS    PRIMARY KEY  (USER_ID)
);

CREATE TABLE PAYMENT_INFO(
   PAYMENT_ID          INTEGER             NOT NULL,
   MEMBERSHIP_NO       VARCHAR2(10)        ,
   CARD_TYPE           VARCHAR2(10)        NOT NULL,
   CARD_NUMBER         INTEGER             NOT NULL UNIQUE,
   CVV                 VARCHAR2(3)         NOT NULL,
   EXPIRY              DATE                NOT NULL,

   CONSTRAINT      PK_PAYMENT_INFO     PRIMARY KEY (PAYMENT_ID),
   CONSTRAINT      FK_PAYMENT_MEMBERSHIP_NO FOREIGN KEY (MEMBERSHIP_NO)
REFERENCES CUSTOMER(MEMBERSHIP_NO)
);

CREATE TABLE ALL_ORDER(
   ORDER_ID            INTEGER             NOT NULL,
   USER_ID             VARCHAR2(30)        NOT NULL,
   IF_ONLINE           NUMBER(1)           NOT NULL CHECK  (IF_ONLINE IN (0,1)),
   STATUS              VARCHAR2(12)        NOT NULL,
   ORDER_DATE          DATE                NOT NULL,
   NO_ITEMS            INTEGER             NOT NULL,
   TOTAL_AMOUNT        INTEGER             NOT NULL,

   REFERENCE_ID        INTEGER             UNIQUE,

   CONSTRAINT  PK_ALL_ORDER_     PRIMARY KEY (ORDER_ID),
   CONSTRAINT  FK_ALL_ORDER_CUSTOMER_ID    FOREIGN KEY (USER_ID) REFERENCES
CUSTOMER
);

CREATE TABLE SHIPMENT(
   SHIPMENT_ID         INTEGER             NOT NULL,
   REFERENCE_ID        INTEGER             NOT NULL,
   SHIPMENT_STATUS  VARCHAR2(12)NOT NULL,
   DATE_SHIPPED        DATE                ,
   TARGET_DELIVERY_DAYS INTEGER            NOT NULL,
   SHIPMENT_VENDOR     VARCHAR2(50)        NOT NULL,

   CONSTRAINT          PK_SHIPMENT    PRIMARY KEY  (SHIPMENT_ID),
   CONSTRAINT FK_SHIPMENT_ORDER_ONLINE FOREIGN KEY (REFERENCE_ID)
REFERENCES ALL_ORDER(REFERENCE_ID)
);

CREATE TABLE ORDER_HAS_PRODUCTS(
    ORDER_ID            INTEGER             NOT NULL REFERENCES ALL_ORDER(ORDER_ID) ,
    PRODUCT_ID          INTEGER             NOT NULL REFERENCES PRODUCT(PRODUCT_ID) ,

    CONSTRAINT PK_ORDER_WITH_PRODUCTS PRIMARY KEY (ORDER_ID, PRODUCT_ID)
);

CREATE TABLE STORE_SELLS_PRODUCTS(
    STORE_ID            INTEGER             NOT NULL REFERENCES STORE_INFO(STORE_ID) ,
    PRODUCT_ID          INTEGER             NOT NULL REFERENCES PRODUCT(PRODUCT_ID) ,

    CONSTRAINT PK_STORE_WITH_PRODUCTS PRIMARY KEY (STORE_ID, PRODUCT_ID)
);

/* CREATING INDICES FOR NATURAL KEYS/FOREIGN KEYS AND FREQUENTLY-QUERIED
COLUMNS */

-- EMPLOYEE
-- FREQUENTLY QUERIED
CREATE INDEX IDX_EMPLOYEE_ZIPCODE ON EMPLOYEE (ZIPCODE);

-- MANUFACTURER
-- FREQUENTLY QUERIED
CREATE INDEX IDX_MANUFACTURER_COUNTRY ON MANUFACTURER (COUNTRY);

-- PRODUCT
-- FOREIGN KEYS
CREATE INDEX IDX_PRODUCT_MANUFACTURER ON PRODUCT (MANUFACTURER_ID);

-- FREQUENTLY QUERIED
CREATE INDEX IDX_PRODUCT_PRICE ON PRODUCT (PRICE);
CREATE INDEX IDX_PRODUCT_INVENTORY ON PRODUCT (INVENTORY);
CREATE INDEX IDX_PRODUCT_TAXABLE ON PRODUCT (TAXABLE);

-- CUSTOMER
-- NATURAL KEYS
CREATE INDEX IDX_CUSTOMER_PHONE ON CUSTOMER (PHONE);

-- FREQUENTLY QUERIED
CREATE INDEX IDX_CUSTOMER_ZIPCODE ON CUSTOMER (ZIPCODE);
CREATE INDEX IDX_CUSTOMER_DOB ON CUSTOMER (DOB);
CREATE INDEX IDX_CUSTOMER_STATE ON CUSTOMER (STATE_LOC);
CREATE INDEX IDX_IF_MEMBER ON CUSTOMER (IF_MEMBER);

-- PAYMENT_INFO
--NATURAL KEYS
-- FREQUENTLY QUERIED
CREATE INDEX IDX_PAYMENT_CARD_TYPE ON PAYMENT_INFO (CARD_TYPE);

-- ALL_ORDER
-- FOREIGN KEYS
CREATE INDEX IDX_ALL_ORDER_USER_ID ON ALL_ORDER(USER_ID);

-- FREQUNTLY QUERIED
CREATE INDEX IDX_ALL_ORDER_STATUS ON ALL_ORDER(STATUS);
CREATE INDEX IDX_ALL_ORDER_DATE ON ALL_ORDER(ORDER_DATE);
CREATE INDEX IDX_ALL_ORDER_ITEMS ON ALL_ORDER(NO_ITEMS);
CREATE INDEX IDX_ALL_ORDER_TOTAL_AMOUNT ON ALL_ORDER(TOTAL_AMOUNT);

-- SHIPMENT
-- FOREING KEYS
CREATE INDEX IDX_SHIPMENT_REFERENCE_ID ON SHIPMENT (REFERENCE_ID);

-- FREQUENTLY QUERIED
CREATE INDEX IDX_SHIPMENT_STATUS ON SHIPMENT (SHIPMENT_STATUS);
CREATE INDEX IDX_SHIPMENT_TARGET_DELIVERY ON SHIPMENT
(TARGET_DELIVERY_DAYS);
CREATE INDEX IDX_SHIPMENT_VENDOR ON SHIPMENT (SHIPMENT_VENDOR);
```

```
/* Alter Tables by adding Audit columns */
ALTER TABLE STORE_INFO ADD (
CREATED_BY VARCHAR2(30), DATE_CREATED DATE, MODIFIED_BY VARCHAR2(30),
DATE_MODIFIED DATE
);

ALTER TABLE EMPLOYEE ADD (
CREATED_BY VARCHAR2(30), DATE_CREATED DATE, MODIFIED_BY VARCHAR2(30),
DATE_MODIFIED DATE
);

ALTER TABLE MANUFACTURER ADD (
CREATED_BY VARCHAR2(30), DATE_CREATED DATE, MODIFIED_BY VARCHAR2(30),
DATE_MODIFIED DATE
);

ALTER TABLE CUSTOMER ADD (
CREATED_BY VARCHAR2(30), DATE_CREATED DATE, MODIFIED_BY VARCHAR2(30),
DATE_MODIFIED DATE
);

ALTER TABLE ALL_ORDER ADD (
CREATED_BY VARCHAR2(30), DATE_CREATED DATE, MODIFIED_BY VARCHAR2(30),
DATE_MODIFIED DATE
);

ALTER TABLE PAYMENT_INFO ADD (
CREATED_BY VARCHAR2(30), DATE_CREATED DATE, MODIFIED_BY VARCHAR2(30),
DATE_MODIFIED DATE
);

ALTER TABLE PRODUCT ADD (
CREATED_BY VARCHAR2(30), DATE_CREATED DATE, MODIFIED_BY VARCHAR2(30),
DATE_MODIFIED DATE
);

ALTER TABLE SHIPMENT ADD (
CREATED_BY VARCHAR2(30), DATE_CREATED DATE, MODIFIED_BY VARCHAR2(30),
DATE_MODIFIED DATE
);

/* CREATING VIEWS */

CREATE OR REPLACE VIEW MEMBERS AS
SELECT USER_ID, IF_MEMBER, FIRST_NAME, MIDDLE, LAST_NAME, DOB, EMAIL, PHONE,
STREET, CITY, ZIPCODE, STATE_LOC, MEMBERSHIP_NO
FROM CUSTOMER
WHERE MEMBERSHIP_NO IS NOT NULL;

CREATE OR REPLACE VIEW NON_MEMBER_CUSTOMERS AS
SELECT USER_ID, IF_MEMBER, FIRST_NAME, MIDDLE, LAST_NAME, DOB, EMAIL, PHONE,
STREET, CITY, ZIPCODE, STATE_LOC
FROM CUSTOMER
WHERE MEMBERSHIP_NO IS NULL;

CREATE OR REPLACE VIEW OFFLINE_ORDER AS
SELECT ORDER_ID, IF_ONLINE, USER_ID, STATUS, ORDER_DATE, NO_ITEMS,
TOTAL_AMOUNT
FROM ALL_ORDER
WHERE REFERENCE_ID IS NULL;

CREATE OR REPLACE VIEW ONLINE_ORDER AS
SELECT ORDER_ID, IF_ONLINE, USER_ID, STATUS, ORDER_DATE, NO_ITEMS,
TOTAL_AMOUNT, REFERENCE_ID
FROM ALL_ORDER
WHERE REFERENCE_ID IS NOT NULL;

/* CREATING SEQUENCES */
```

```
CREATE SEQUENCE SEQ_STORE_ID
   INCREMENT BY 1
   START WITH 10000
   NOMAXVALUE
   MINVALUE 10000
   NOCACHE;
CREATE SEQUENCE SEQ_EMPLOYEE_ID
   INCREMENT BY 1
   START WITH 1000000
   NOMAXVALUE
   MINVALUE 0
   NOCACHE;
CREATE SEQUENCE SEQ_MANUFACTURER_ID
   INCREMENT BY 1
   START WITH 100000
   NOMAXVALUE
   MINVALUE 100
   NOCACHE;
CREATE SEQUENCE SEQ_PRODUCT_ID
   INCREMENT BY 1
   START WITH 100000
   NOMAXVALUE
   MINVALUE 0
   NOCACHE;
CREATE SEQUENCE SEQ_MEMBERSHIP_NO
   INCREMENT BY 1
   START WITH 10000000
   NOMAXVALUE
   MINVALUE 0
   NOCACHE;
CREATE SEQUENCE SEQ_ORDER_ID
   INCREMENT BY 1
   START WITH 1000000000
   NOMAXVALUE
```

```
    MINVALUE 0
    NOCACHE;
CREATE SEQUENCE SEQ_REFERENCE_ID
    INCREMENT BY 1
    START WITH 1000000000
    NOMAXVALUE
    MINVALUE 0
    NOCACHE;
CREATE SEQUENCE SEQ_PAYMENT_ID
    INCREMENT BY 1
    START WITH 1000000000
    NOMAXVALUE
    MINVALUE 0
    NOCACHE;
CREATE SEQUENCE SEQ_SHIPMENT_ID
    INCREMENT BY 1
    START WITH 1000000000
    NOMAXVALUE
    MINVALUE 0
    NOCACHE;
```

```
/* CREATING TRIGGERS */

-- Business purpose: The TRG_STORE trigger will automatically assign a new sequential STORE_ID to the row whenever a new row is inserted in the STORE_INFO table. It will also assign appropriate values in the created by and created date
-- when a new row is created and appropriate values for modified by and modified date when a row is updated.
```

```
CREATE OR REPLACE TRIGGER TRG_STORE
  BEFORE INSERT OR UPDATE ON STORE_INFO
  FOR EACH ROW
  BEGIN
    IF INSERTING THEN
      IF :NEW.STORE_ID IS NULL THEN
        :NEW.STORE_ID := SEQ_STORE_ID.NEXTVAL;
      END IF;
      IF :NEW.CREATED_BY IS NULL THEN
        :NEW.CREATED_BY := USER;
      END IF;
      IF :NEW.DATE_CREATED IS NULL THEN
        :NEW.DATE_CREATED := SYSDATE;
      END IF;
    END IF;
    IF INSERTING OR UPDATING THEN
      :NEW.MODIFIED_BY := USER;
      :NEW.DATE_MODIFIED := SYSDATE;
    END IF;
  END;
/
```

-- Business purpose: The TRG_EMPLOYEE trigger will automatically assign a new sequential EMPLOYEE_ID to the row whenever a new row is inserted in the EMPLOYEE table. It will also assign appropriate values in the created by and created date
-- when a new row is created and appropriate values for modified by and modified date when a row is updated.

```
CREATE OR REPLACE TRIGGER TRG_EMPLOYEE
  BEFORE INSERT OR UPDATE ON EMPLOYEE
  FOR EACH ROW
  BEGIN
    IF INSERTING THEN
      IF :NEW.EMPLOYEE_ID IS NULL THEN
```

```
            :NEW.EMPLOYEE_ID := SEQ_EMPLOYEE_ID.NEXTVAL;
        END IF;
        IF :NEW.CREATED_BY IS NULL THEN
            :NEW.CREATED_BY := USER;
        END IF;
        IF :NEW.DATE_CREATED IS NULL THEN
            :NEW.DATE_CREATED := SYSDATE;
        END IF;
    END IF;
    IF INSERTING OR UPDATING THEN
        :NEW.MODIFIED_BY := USER;
        :NEW.DATE_MODIFIED := SYSDATE;
    END IF;
  END;
/
```

-- Business purpose: The TRG_MANUFACTURER trigger will automatically assign a new sequential MANUFACTURER_ID to the row whenever a new row is inserted in the MANUFACTURER table. It will also assign appropriate values in the created by and created date
-- when a new row is created and appropriate values for modified by and modified date when a row is updated.

```
CREATE OR REPLACE TRIGGER TRG_MANUFACTURER
  BEFORE INSERT OR UPDATE ON MANUFACTURER
  FOR EACH ROW
  BEGIN
    IF INSERTING THEN
        IF :NEW.MANUFACTURER_ID IS NULL THEN
            :NEW.MANUFACTURER_ID := SEQ_MANUFACTURER_ID.NEXTVAL;
        END IF;
        IF :NEW.CREATED_BY IS NULL THEN
            :NEW.CREATED_BY := USER;
        END IF;
        IF :NEW.DATE_CREATED IS NULL THEN
```

```
            :NEW.DATE_CREATED := SYSDATE;
        END IF;
    END IF;
    IF INSERTING OR UPDATING THEN
        :NEW.MODIFIED_BY := USER;
        :NEW.DATE_MODIFIED := SYSDATE;
    END IF;
END;
/
```

-- Business purpose: The TRG_PRODUCT trigger will automatically assign a new sequential PRODUCT_ID to the row whenever a new row is inserted in the PRODUCT table. It will also assign appropriate values in the created by and created date
-- when a new row is created and appropriate values for modified by and modified date when a row is updated.

```
CREATE OR REPLACE TRIGGER TRG_PRODUCT
    BEFORE INSERT OR UPDATE ON PRODUCT
    FOR EACH ROW
    BEGIN
        IF INSERTING THEN
            IF :NEW.PRODUCT_ID IS NULL THEN
                :NEW.PRODUCT_ID := SEQ_PRODUCT_ID.NEXTVAL;
            END IF;
            IF :NEW.CREATED_BY IS NULL THEN
                :NEW.CREATED_BY := USER;
            END IF;
            IF :NEW.DATE_CREATED IS NULL THEN
                :NEW.DATE_CREATED := SYSDATE;
            END IF;
        END IF;
        IF INSERTING OR UPDATING THEN
            :NEW.MODIFIED_BY := USER;
            :NEW.DATE_MODIFIED := SYSDATE;
        END IF;
    END;
/
```

-- Business purpose: The TRG_CUSTOMER trigger will automatically assign a new sequential CUSTOMER_ID to the row whenever a new row is inserted in the CUSTOMER table. It will also assign a new MEMBERSHIP_NO to the row whenever a new row with
-- IF_MEMBER column member as TRUE is encountered. It will also assign appropriate values in the created by and created date when a new row is created and appropriate values for modified by and modified date when a row is updated.

```
CREATE OR REPLACE TRIGGER TRG_CUSTOMER
    BEFORE INSERT OR UPDATE ON CUSTOMER
    FOR EACH ROW
    BEGIN
        IF INSERTING THEN
            IF : NEW.MEMBERSHIP_NO IS NULL THEN
                IF : NEW.IF_MEMBER IN (1) THEN
                    :NEW.MEMBERSHIP_NO := SEQ_REFERENCE_ID.NEXTVAL;
                END IF;
            END IF;
            IF :NEW.CREATED_BY IS NULL THEN
                :NEW.CREATED_BY := USER;
            END IF;
            IF :NEW.DATE_CREATED IS NULL THEN
                :NEW.DATE_CREATED := SYSDATE;
            END IF;
        END IF;
        IF INSERTING OR UPDATING THEN
            :NEW.MODIFIED_BY := USER;
            :NEW.DATE_MODIFIED := SYSDATE;
        END IF;
    END;
/
```

-- Business purpose: The TRG_ORDER trigger will automatically assign a new sequential ORDER_ID to the row whenever a new row is inserted in the ORDER table. It will also assign a new REFERENCE_ID to the row whenever a new row with
-- IF_ONLINE column is encountered as TRUE. It will also assign appropriate values in the created by and created date when a new row is created and appropriate values for modified by and modified date when a row is updated.

```
CREATE OR REPLACE TRIGGER TRG_ORDER
  BEFORE INSERT OR UPDATE ON ALL_ORDER
  FOR EACH ROW
  BEGIN
    IF INSERTING THEN
      IF :NEW.ORDER_ID IS NULL THEN
        :NEW.ORDER_ID := SEQ_ORDER_ID.NEXTVAL;
      END IF;
      IF : NEW.REFERENCE_ID IS NULL THEN
        IF : NEW.IF_ONLINE IN (1) THEN
          :NEW.REFERENCE_ID := SEQ_REFERENCE_ID.NEXTVAL;
        END IF;
      END IF;
      IF :NEW.CREATED_BY IS NULL THEN
        :NEW.CREATED_BY := USER;
      END IF;
      IF :NEW.DATE_CREATED IS NULL THEN
        :NEW.DATE_CREATED := SYSDATE;
      END IF;
    END IF;
    IF INSERTING OR UPDATING THEN
      :NEW.MODIFIED_BY := USER;
      :NEW.DATE_MODIFIED := SYSDATE;
    END IF;
  END;
/
```

-- Business purpose: The TRG_PAYMENT trigger will automatically assign a new sequential PAYMENT_ID to the row whenever a new row is inserted in the PAYMENT table. It will also assign appropriate values in the created by and created date
-- when a new row is created and appropriate values for modified by and modified date when a row is updated.

```
CREATE OR REPLACE TRIGGER TRG_PAYMENT
  BEFORE INSERT OR UPDATE ON PAYMENT_INFO
  FOR EACH ROW
  BEGIN
    IF INSERTING THEN
      IF :NEW.PAYMENT_ID IS NULL THEN
        :NEW.PAYMENT_ID := SEQ_PAYMENT_ID.NEXTVAL;
      END IF;
      IF :NEW.CREATED_BY IS NULL THEN
        :NEW.CREATED_BY := USER;
      END IF;
      IF :NEW.DATE_CREATED IS NULL THEN
        :NEW.DATE_CREATED := SYSDATE;
      END IF;
    END IF;
    IF INSERTING OR UPDATING THEN
      :NEW.MODIFIED_BY := USER;
      :NEW.DATE_MODIFIED := SYSDATE;
    END IF;
  END;
/
```

-- Business purpose: The TRG_SHIPMENT trigger will automatically assign a new sequential SHIPMENT_ID to the row whenever a new row is inserted in the SHIPMENT table. It will also assign appropriate values in the created by and created date
-- when a new row is created and appropriate values for modified by and modified date when a row is updated.

```
CREATE OR REPLACE TRIGGER TRG_SHIPMENT
  BEFORE INSERT OR UPDATE ON SHIPMENT
  FOR EACH ROW
  BEGIN
    IF INSERTING THEN
      IF :NEW.SHIPMENT_ID IS NULL THEN
        :NEW.SHIPMENT_ID := SEQ_SHIPMENT_ID.NEXTVAL;
      END IF;
      IF :NEW.CREATED_BY IS NULL THEN
        :NEW.CREATED_BY := USER;
      END IF;
      IF :NEW.DATE_CREATED IS NULL THEN
        :NEW.DATE_CREATED := SYSDATE;
      END IF;
    END IF;
    IF INSERTING OR UPDATING THEN
      :NEW.MODIFIED_BY := USER;
      :NEW.DATE_MODIFIED := SYSDATE;
    END IF;
  END;
/
```

DML AND QUERY SOURCE CODE

```
/* Populate all tables */
-- STORE_INFO
INSERT INTO STORE_INFO ( STORE_LOC, STREET, CITY, STATE_NAME, AREA_IN_SQFT,
LIQUOR_SALES)
VALUES ('East Village', '123 Road Rd', 'Dallas', 'TX', 42555, 0);
INSERT INTO STORE_INFO ( STORE_LOC, STREET, CITY, STATE_NAME, AREA_IN_SQFT,
LIQUOR_SALES)
```

```
VALUES ('West Villas', '456 Gram Rd', 'Jolla', 'CA', 76391, 1);
INSERT INTO STORE_INFO (STORE_LOC, STREET, CITY, STATE_NAME, AREA_IN_SQFT,
LIQUOR_SALES)
VALUES ('Great Mall', '123 Road Rd', 'Dallas', 'TX', 74927, 0);
INSERT INTO STORE_INFO ( STORE_LOC, STREET, CITY, STATE_NAME, AREA_IN_SQFT,
LIQUOR_SALES)
VALUES ('Galleria', '839 North St', 'Houston', 'TX', 72946, 1);
INSERT INTO STORE_INFO ( STORE_LOC, STREET, CITY, STATE_NAME, AREA_IN_SQFT,
LIQUOR_SALES)
VALUES ( 'Parks at Rec', '382 Lest Dr', 'Denver', 'CO', 73629, 0);
INSERT INTO STORE_INFO ( STORE_LOC, STREET, CITY, STATE_NAME, AREA_IN_SQFT,
LIQUOR_SALES)
VALUES ( 'North Plaza', '735 Tech Dr', 'Kent', 'MN', 83629, 1);
INSERT INTO STORE_INFO ( STORE_LOC, STREET, CITY, STATE_NAME, AREA_IN_SQFT,
LIQUOR_SALES)
VALUES ( 'Park West', '138 Crest Rd', 'Dallas', 'TX', 38163, 0);
INSERT INTO STORE_INFO ( STORE_LOC, STREET, CITY, STATE_NAME, AREA_IN_SQFT,
LIQUOR_SALES)
VALUES ( 'Faster Way', '3729 State St', 'Ten', 'IL', 73610, 1 );
INSERT INTO STORE_INFO ( STORE_LOC, STREET, CITY, STATE_NAME, AREA_IN_SQFT,
LIQUOR_SALES)
VALUES ( 'West Tent', '271 Dog Dr', 'Houston', 'TX', 37193, 1);
INSERT INTO STORE_INFO ( STORE_LOC, STREET, CITY, STATE_NAME, AREA_IN_SQFT,
LIQUOR_SALES)
VALUES ( 'Magic West', '382 Magic Rd',  'Austin',  'TX', 74629, 0);
INSERT INTO STORE_INFO ( STORE_LOC, STREET, CITY, STATE_NAME, AREA_IN_SQFT,
LIQUOR_SALES)
VALUES ( 'Great Express', '3719 Track St', 'Austin',  'TX', 84623, 1);
INSERT INTO STORE_INFO ( STORE_LOC, STREET, CITY, STATE_NAME, AREA_IN_SQFT,
LIQUOR_SALES)
VALUES ( 'Lovers Field', '328 Pix Wy',  'Dallas',  'TX', 37153, 0);
INSERT INTO STORE_INFO ( STORE_LOC, STREET, CITY, STATE_NAME, AREA_IN_SQFT,
LIQUOR_SALES)
VALUES ( 'Lake West', '471 Lake Dr', 'Denver',  'CO', 47293, 1);
```

```
INSERT INTO STORE_INFO ( STORE_LOC, STREET, CITY, STATE_NAME, AREA_IN_SQFT,
LIQUOR_SALES)
VALUES ( 'Pearl Plaza', '3725 Frankford Rd', 'Dallas',  'TX', 47294, 0);
INSERT INTO STORE_INFO ( STORE_LOC, STREET, CITY, STATE_NAME, AREA_IN_SQFT,
LIQUOR_SALES)
VALUES ( 'Right Park', '358 Parkway Dr',  'Houston',  'TX', 37258, 1);

-- EMPLOYEE
INSERT INTO EMPLOYEE ( STORE_ID, FIRST_NAME, MIDDLE, LAST_NAME,
DATE_EMPLOYED, SSN, DOB, EMAIL, PHONE, STREET, CITY, ZIPCODE,  STATE_LOC)
VALUES ( 10000, 'John', 'M',  'Doe', TO_DATE('01/01/2022', 'MM/DD/YYYY'), 123456789,
TO_DATE('10/11/2000', 'MM/DD/YYYY'),  'johndoe@gmail.com', '123-456-7892', '123 ABC St',
'Dallas', '12345', 'TX');
INSERT INTO EMPLOYEE ( STORE_ID, FIRST_NAME, MIDDLE, LAST_NAME,
DATE_EMPLOYED, SSN, DOB, EMAIL, PHONE, STREET, CITY, ZIPCODE,  STATE_LOC)
VALUES ( 10002, 'Rachel', 'H',  'Pham', TO_DATE('01/05/2022', 'MM/DD/YYYY'), 125928361,
TO_DATE('05/10/2000', 'MM/DD/YYYY'), 'rachelpham@gmail.com', '134-123-7842', '432 DFE St',
'Houston', '82491', 'TX');
INSERT INTO EMPLOYEE ( STORE_ID, FIRST_NAME, MIDDLE, LAST_NAME,
DATE_EMPLOYED, SSN, DOB, EMAIL, PHONE, STREET, CITY, ZIPCODE,  STATE_LOC)
VALUES ( 10002, 'Ben', 'M',  'Stripe', TO_DATE('02/01/2022', 'MM/DD/YYYY'), 837291730,
TO_DATE('10/04/1992', 'MM/DD/YYYY'),  'benstripe@gmail.com', '821-456-7391', '193 Ben St',
'Dallas', '74920', 'TX');
INSERT INTO EMPLOYEE ( STORE_ID, FIRST_NAME, MIDDLE, LAST_NAME,
DATE_EMPLOYED, SSN, DOB, EMAIL, PHONE, STREET, CITY, ZIPCODE,  STATE_LOC)
VALUES ( 10003,  'Kayla', 'M',  'Men', TO_DATE('09/01/2022' ,'MM/DD/YYYY'), 738192836,
TO_DATE('04/10/1970', 'MM/DD/YYYY'),  'kaylamen@gmail.com', '241-421-3816', '8391 Great St',
'Denver', '82018', 'CO');
INSERT INTO EMPLOYEE ( STORE_ID, FIRST_NAME, MIDDLE, LAST_NAME,
DATE_EMPLOYED, SSN, DOB, EMAIL, PHONE, STREET, CITY, ZIPCODE,  STATE_LOC)
VALUES ( 10004,  'Jenn', 'T',  'Ester', TO_DATE('02/05/2022', 'MM/DD/YYYY'),  374627382,
TO_DATE('09/05/1980', 'MM/DD/YYYY'), 'jennester@gmail.com', '693-242-2453', '228 Nice St', 'San
Antonio', '38193', 'TX');
```

```
INSERT INTO EMPLOYEE ( STORE_ID, FIRST_NAME, MIDDLE, LAST_NAME,
DATE_EMPLOYED, SSN, DOB, EMAIL, PHONE, STREET, CITY, ZIPCODE,  STATE_LOC)
VALUES ( 10005, 'Nelly', 'G',  'Zoe', TO_DATE('10/02/2022', 'MM/DD/YYYY'), 347824820,
TO_DATE('02/15/1995', 'MM/DD/YYYY'),  'nellyzoe@gmail.com', '472-425-8271', '153 ABC St',
'Dallas', '12345', 'TX');
INSERT INTO EMPLOYEE ( STORE_ID, FIRST_NAME, MIDDLE, LAST_NAME,
DATE_EMPLOYED, SSN, DOB, EMAIL, PHONE, STREET, CITY, ZIPCODE,  STATE_LOC)
VALUES ( 10006,  'Mitchell', 'S',  'Dock', TO_DATE('07/02/2022', 'MM/DD/YYYY'),  281048246,
TO_DATE('07/10/1992', 'MM/DD/YYYY'), 'mitchelldock@gmail.com', '739-264-2716', '672 Alpha St',
'San Antonio', '38193', 'TX');
INSERT INTO EMPLOYEE ( STORE_ID, FIRST_NAME, MIDDLE, LAST_NAME,
DATE_EMPLOYED, SSN, DOB, EMAIL, PHONE, STREET, CITY, ZIPCODE,  STATE_LOC)
VALUES ( 10007,  'Michael', 'S',  'Smith', TO_DATE('10/20/2022', 'MM/DD/YYYY'), 736517251,
TO_DATE('03/28/1995',  'MM/DD/YYYY'), 'michaelsmitch@aol.com', '846-742-7261', '568 Ten St',
'Dallas', '12345', 'TX');
INSERT INTO EMPLOYEE ( STORE_ID, FIRST_NAME, MIDDLE, LAST_NAME,
DATE_EMPLOYED, SSN, DOB, EMAIL, PHONE, STREET, CITY, ZIPCODE,  STATE_LOC)
VALUES ( 10002, 'Riley', 'E',  'Johnson', TO_DATE('05/10/2022', 'MM/DD/YYYY'), 735183972,
TO_DATE('11/02/1982', 'MM/DD/YYYY'), 'rileyjohnson@gmail.com', '837-246-7362', '9361 General
St', 'Houston', '72519', 'TX');
INSERT INTO EMPLOYEE ( STORE_ID, FIRST_NAME, MIDDLE, LAST_NAME,
DATE_EMPLOYED, SSN, DOB, EMAIL, PHONE, STREET, CITY, ZIPCODE,  STATE_LOC)
VALUES ( 10002,  'Alexa', 'M',  'Nguyen', TO_DATE('11/05/2022', 'MM/DD/YYYY'), 371682619,
TO_DATE('12/11/1985', 'MM/DD/YYYY'), 'alexanguyen@yahoo.com', '836-271-4678', '678 XYZ St',
'Denver', 82018, 'CO');
INSERT INTO EMPLOYEE ( STORE_ID, FIRST_NAME, MIDDLE, LAST_NAME,
DATE_EMPLOYED, SSN, DOB, EMAIL, PHONE, STREET, CITY, ZIPCODE,  STATE_LOC)
VALUES ( 10000,  'Joseph', 'A',  'Lee', TO_DATE('08/19/2022', 'MM/DD/YYYY'), 625172539,
TO_DATE('02/19/1979', 'MM/DD/YYYY'), 'josephlee@gmail.com', '749-361-3759', '892 Bentley Rd',
'Dallas', 83018, 'TX');
INSERT INTO EMPLOYEE ( STORE_ID, FIRST_NAME, MIDDLE, LAST_NAME,
DATE_EMPLOYED, SSN, DOB, EMAIL, PHONE, STREET, CITY, ZIPCODE,  STATE_LOC)
```

```
VALUES ( 10003,  'Joey', 'I',  'Mest', TO_DATE('09/19/2022', 'MM/DD/YYYY'), 746294726,
TO_DATE('10/09/1994', 'MM/DD/YYYY'), 'joeymest@gmail.com', '472-462-4728', '345 Har St',
'Houston', 47294, 'TX');
INSERT INTO EMPLOYEE ( STORE_ID, FIRST_NAME, MIDDLE, LAST_NAME,
DATE_EMPLOYED, SSN, DOB, EMAIL, PHONE, STREET, CITY, ZIPCODE,  STATE_LOC)
VALUES ( 10004,  'Kayla', 'K',  'Mendoza', TO_DATE('06/10/2022', 'MM/DD/YYYY'), 726193749,
TO_DATE('11/08/1992', 'MM/DD/YYYY'),  'kaylamendoza@gmail.com', '846-371-7482', '1010 Tenth
St', 'Dallas', 38173, 'TX');
INSERT INTO EMPLOYEE ( STORE_ID, FIRST_NAME, MIDDLE, LAST_NAME,
DATE_EMPLOYED, SSN, DOB, EMAIL, PHONE, STREET, CITY, ZIPCODE,  STATE_LOC)
VALUES ( 10005,  'Lexi', 'A',  'Riley', TO_DATE('08/10/2022', 'MM/DD/YYYY'), 4728391729,
TO_DATE('03/24/2002', 'MM/DD/YYYY'),  'lexiriley@gmail.com', '846-826-7492', '1198 Hundred St',
'Austin', 39174, 'TX');
INSERT INTO EMPLOYEE ( STORE_ID, FIRST_NAME, MIDDLE, LAST_NAME,
DATE_EMPLOYED, SSN, DOB, EMAIL, PHONE, STREET, CITY, ZIPCODE,  STATE_LOC)
VALUES ( 10006, 'Josh', 'P',  'Don', TO_DATE('03/22/2022', 'MM/DD/YYYY'), 2735193715,
TO_DATE('09/21/1988', 'MM/DD/YYYY'), 'joshdon@gmail.com', '172-735-8715', '1986 Trail St',
'Austin', 47225, 'TX');

-- CUSTOMER
INSERT INTO CUSTOMER (CUSTOMER_ID, IF_MEMBER, FIRST_NAME, MIDDLE,
LAST_NAME, DOB, EMAIL, PHONE, STREET, CITY, ZIPCODE,  STATE_LOC)
VALUES ('RACHELSHOPS19', 0, 'Rachel', 'H', 'Pham', TO_DATE('09/05/1965',  'MM/DD/YYYY'),
'rachelpham@gmail.com', 4694283574, '476 Grand Dr', 'Dallas', 63820, 'TX');
INSERT INTO CUSTOMER (CUSTOMER_ID, IF_MEMBER, FIRST_NAME, MIDDLE,
LAST_NAME, DOB, EMAIL, PHONE, STREET, CITY, ZIPCODE,  STATE_LOC)
VALUES ('MIHIR4766', 1, 'Mihir', 'T', 'Harvi', TO_DATE('10/25/1975',  'MM/DD/YYYY'),
'mihir@gmail.com', 2313213123,'1236 Next Dr', 'Houston', 47639, 'TX');
INSERT INTO CUSTOMER (CUSTOMER_ID, IF_MEMBER, FIRST_NAME, MIDDLE,
LAST_NAME, DOB, EMAIL, PHONE, STREET, CITY, ZIPCODE,  STATE_LOC)
VALUES ('KAYLA7203', 0, 'Kayla', 'H', 'Smith', TO_DATE('04/15/1995',  'MM/DD/YYYY'),
'kayla@gmail.com', 4445556986, '739 May Dr', 'Dallas', 68490, 'TX');
INSERT INTO CUSTOMER (CUSTOMER_ID, IF_MEMBER, FIRST_NAME, MIDDLE,
LAST_NAME, DOB, EMAIL, PHONE, STREET, CITY, ZIPCODE,  STATE_LOC)
```

```
VALUES ('RILES9466', 1, 'Riley', 'E', 'Frost', TO_DATE('11/28/1997',  'MM/DD/YYYY'),
'riley@gmail.com', 1238870962, '276 Lake Dr', 'Chicago', 47639, 'IL');
INSERT INTO CUSTOMER (CUSTOMER_ID, IF_MEMBER, FIRST_NAME, MIDDLE,
LAST_NAME, DOB, EMAIL, PHONE, STREET, CITY, ZIPCODE,  STATE_LOC)
VALUES ('JODIE86875', 0, 'Jodie', 'H', 'Pham', TO_DATE('08/05/2005',  'MM/DD/YYYY'),
'jodie@gmail.com', 4242447898, '228 Mister Rd', 'Denver', 73583, 'CO');
INSERT INTO CUSTOMER (CUSTOMER_ID, IF_MEMBER, FIRST_NAME, MIDDLE,
LAST_NAME, DOB, EMAIL, PHONE, STREET, CITY, ZIPCODE,  STATE_LOC)
VALUES ('LOVESMITH3869', 0, 'Love', 'Y', 'Smith', TO_DATE('10/06/1955',  'MM/DD/YYYY'),
'lovesmith@gmail.com', 2223334675, '4785 West Way', 'Dallas', 63820, 'TX');
INSERT INTO CUSTOMER (CUSTOMER_ID, IF_MEMBER, FIRST_NAME, MIDDLE,
LAST_NAME, DOB, EMAIL, PHONE, STREET, CITY, ZIPCODE,  STATE_LOC)
VALUES ('BRUCELEE8379', 1, 'Bruce', 'H', 'Lee', TO_DATE('12/18/2001',  'MM/DD/YYYY'),
'brucelee@gmail.com', 2342227878, '234 Lux Dr', 'Houston', 47630, 'TX');
INSERT INTO CUSTOMER (CUSTOMER_ID, IF_MEMBER, FIRST_NAME, MIDDLE,
LAST_NAME, DOB, EMAIL, PHONE, STREET, CITY, ZIPCODE,  STATE_LOC)
VALUES ('BUAN6320', 0, 'Gasan', 'X', 'Elkhodari', TO_DATE('09/05/1995',  'MM/DD/YYYY'),
'gelkhodari@gmail.com', 2314343231, '123 Prof Ln', 'Dallas', 63820, 'TX');
INSERT INTO CUSTOMER (CUSTOMER_ID, IF_MEMBER, FIRST_NAME, MIDDLE,
LAST_NAME, DOB, EMAIL, PHONE, STREET, CITY, ZIPCODE,  STATE_LOC)
VALUES ('BENTLEYSHOPPER27', 1, 'Bentley', 'H', 'John', TO_DATE('08/15/1969',  'MM/DD/YYYY'),
'bentley@gmail.com', 1212334213, '434 Dog Dr', 'Frisco', 74620, 'TX');
INSERT INTO CUSTOMER (CUSTOMER_ID, IF_MEMBER, FIRST_NAME, MIDDLE,
LAST_NAME, DOB, EMAIL, PHONE, STREET, CITY, ZIPCODE,  STATE_LOC)
VALUES ('DAVEATS736', 0, 'Dave', 'P', 'Treck', TO_DATE('12/11/1965',  'MM/DD/YYYY'),
'dave@gmail.com', 2323124421, '365 Lakey Dr', 'Denver', 32768, 'CO');
INSERT INTO CUSTOMER (CUSTOMER_ID, IF_MEMBER, FIRST_NAME, MIDDLE,
LAST_NAME, DOB, EMAIL, PHONE, STREET, CITY, ZIPCODE,  STATE_LOC)
VALUES ('TIMBERLAND476', 1, 'Tim', 'L', 'Lester', TO_DATE('03/30/1977',  'MM/DD/YYYY'),
'timberland@gmail.com', 9094238842, '375 Forest Wy', 'Chicago', 73582, 'IL');
INSERT INTO CUSTOMER (CUSTOMER_ID, IF_MEMBER, FIRST_NAME, MIDDLE,
LAST_NAME, DOB, EMAIL, PHONE, STREET, CITY, ZIPCODE,  STATE_LOC)
VALUES ('SHOPLOVER378', 0, 'Tina', 'T', 'West', TO_DATE('11/16/1984',  'MM/DD/YYYY'),
'tinaw@gmail.com', 1231232245, '478 E HW', 'Austin', 47899, 'TX');
```

```
INSERT INTO CUSTOMER (CUSTOMER_ID, IF_MEMBER, FIRST_NAME, MIDDLE,
LAST_NAME, DOB, EMAIL, PHONE, STREET, CITY, ZIPCODE,  STATE_LOC)
VALUES ('TENSMITH', 1, 'Michelle', 'H', 'Ten', TO_DATE('08/31/1945',  'MM/DD/YYYY'),
'michelle@gmail.com', 9425678331, '378 Tenth St', 'Allen', 37659, 'TX');
INSERT INTO CUSTOMER (CUSTOMER_ID, IF_MEMBER, FIRST_NAME, MIDDLE,
LAST_NAME, DOB, EMAIL, PHONE, STREET, CITY, ZIPCODE,  STATE_LOC)
VALUES ('TOYTEN386', 0, 'Trent', 'P', 'Grayson', TO_DATE('06/12/1965',  'MM/DD/YYYY'),
'trent@gmail.com', 9981292238, '4759 Briar Way', 'Dallas', 64762, 'TX');
INSERT INTO CUSTOMER (CUSTOMER_ID, IF_MEMBER, FIRST_NAME, MIDDLE,
LAST_NAME, DOB, EMAIL, PHONE, STREET, CITY, ZIPCODE,  STATE_LOC)
VALUES ('FOREVERSHOP3785', 1, 'Brandon', 'P', 'Collin', TO_DATE('05/27/1999',  'MM/DD/YYYY'),
'brandon@gmail.com', 5699820877, '4294 Coyne St', 'San Jose', 47683, 'CA');

-- ALL_ORDER

INSERT INTO ALL_ORDER ( IF_ONLINE, STATUS, ORDER_DATE, NO_ITEMS,
TOTAL_AMOUNT, CUSTOMER_ID)
VALUES ( 1, 'PENDING', TO_DATE('01/19/2022', 'MM/DD/YYYY'), 3, 24.95, 'MIHIR4766');
INSERT INTO ALL_ORDER ( IF_ONLINE, STATUS, ORDER_DATE, NO_ITEMS,
TOTAL_AMOUNT, CUSTOMER_ID)
VALUES ( 0, 'PROCESSING', TO_DATE('03/15/2022', 'MM/DD/YYYY'), 1, 4.95, 'KAYLA7203');
INSERT INTO ALL_ORDER ( IF_ONLINE, STATUS, ORDER_DATE, NO_ITEMS,
TOTAL_AMOUNT, CUSTOMER_ID)
VALUES ( 0, 'SHIPPED', TO_DATE('10/19/2022', 'MM/DD/YYYY'), 15, 194.95, 'RILES9466');
INSERT INTO ALL_ORDER ( IF_ONLINE, STATUS, ORDER_DATE, NO_ITEMS,
TOTAL_AMOUNT, CUSTOMER_ID)
VALUES ( 1, 'PROCESSING', TO_DATE('11/09/2022', 'MM/DD/YYYY'), 5, 54.75, 'JODIE86875');
INSERT INTO ALL_ORDER ( IF_ONLINE, STATUS, ORDER_DATE, NO_ITEMS,
TOTAL_AMOUNT, CUSTOMER_ID)
VALUES ( 1, 'SHIPPED', TO_DATE('04/15/2022', 'MM/DD/YYYY'), 1, 74.95, 'LOVESMITH3869');
INSERT INTO ALL_ORDER ( IF_ONLINE, STATUS, ORDER_DATE, NO_ITEMS,
TOTAL_AMOUNT, CUSTOMER_ID)
VALUES ( 0, 'PROCESSING', TO_DATE('10/19/2022', 'MM/DD/YYYY'), 19, 188.95,
'BRUCELEE8379');
```

```
INSERT INTO ALL_ORDER ( IF_ONLINE, STATUS, ORDER_DATE, NO_ITEMS,
TOTAL_AMOUNT, CUSTOMER_ID)
VALUES ( 1, 'SHIPPED', TO_DATE('03/12/2022', 'MM/DD/YYYY'), 5, 18.95,
'BENTLEYSHOPPER27');
INSERT INTO ALL_ORDER ( IF_ONLINE, STATUS, ORDER_DATE, NO_ITEMS,
TOTAL_AMOUNT, CUSTOMER_ID)
VALUES ( 0, 'PENDING', TO_DATE('07/22/2022', 'MM/DD/YYYY'), 3, 24.95, 'TIMBERLAND476');
INSERT INTO ALL_ORDER ( IF_ONLINE, STATUS, ORDER_DATE, NO_ITEMS,
TOTAL_AMOUNT, CUSTOMER_ID)
VALUES ( 1, 'PROCESSING', TO_DATE('01/19/2022', 'MM/DD/YYYY'), 6, 84.95, 'TENSMITH');
INSERT INTO ALL_ORDER ( IF_ONLINE, STATUS, ORDER_DATE, NO_ITEMS,
TOTAL_AMOUNT, CUSTOMER_ID)
VALUES ( 0, 'SHIPPED', TO_DATE('09/29/2022', 'MM/DD/YYYY'), 6, 94.95, 'TOYTEN386');
INSERT INTO ALL_ORDER ( IF_ONLINE, STATUS, ORDER_DATE, NO_ITEMS,
TOTAL_AMOUNT, CUSTOMER_ID)
VALUES ( 1, 'PENDING', TO_DATE('11/09/2022', 'MM/DD/YYYY'), 37, 1024.95,
'FOREVERSHOP3785');
INSERT INTO ALL_ORDER ( IF_ONLINE, STATUS, ORDER_DATE, NO_ITEMS,
TOTAL_AMOUNT, CUSTOMER_ID)
VALUES ( 1, 'PROCESSING', TO_DATE('10/09/2022', 'MM/DD/YYYY'), 23, 976.85,
'RACHELSHOPS19');
INSERT INTO ALL_ORDER ( IF_ONLINE, STATUS, ORDER_DATE, NO_ITEMS,
TOTAL_AMOUNT, CUSTOMER_ID)
VALUES ( 0, 'SHIPPED', TO_DATE('05/29/2022', 'MM/DD/YYYY'), 7, 96.95, 'MIHIR4766');
INSERT INTO ALL_ORDER ( IF_ONLINE, STATUS, ORDER_DATE, NO_ITEMS,
TOTAL_AMOUNT, CUSTOMER_ID)
VALUES ( 1, 'PENDING', TO_DATE('09/05/2022', 'MM/DD/YYYY'), 3, 24.95, 'MIHIR4766');
INSERT INTO ALL_ORDER ( IF_ONLINE, STATUS, ORDER_DATE, NO_ITEMS,
TOTAL_AMOUNT, CUSTOMER_ID)
VALUES ( 0, 'PROCESSING', TO_DATE('05/09/2022', 'MM/DD/YYYY'), 10, 750.85,
'RACHELSHOPS19');

-- SHIPMENT
```

```
INSERT INTO SHIPMENT ( SHIPMENT_STATUS, DATE_SHIPPED, TARGET_DELIVERY_DAYS,
SHIPMENT_VENDOR, REFERENCE_ID)
VALUES ('DELIVERED', TO_DATE('02/17/2022', 'MM/DD/YYYY'), 5, 'USPS' , 1000000007);
INSERT INTO SHIPMENT ( SHIPMENT_STATUS, DATE_SHIPPED, TARGET_DELIVERY_DAYS,
SHIPMENT_VENDOR, REFERENCE_ID)
VALUES ('SHIPPED', TO_DATE('05/17/2022', 'MM/DD/YYYY'), 6, 'USPS', 1000000008);
INSERT INTO SHIPMENT ( SHIPMENT_STATUS, DATE_SHIPPED, TARGET_DELIVERY_DAYS,
SHIPMENT_VENDOR, REFERENCE_ID)
VALUES ('DELIVERED', TO_DATE('02/22/2022', 'MM/DD/YYYY'), 5, 'FEDEX', 1000000009);
INSERT INTO SHIPMENT ( SHIPMENT_STATUS, DATE_SHIPPED, TARGET_DELIVERY_DAYS,
SHIPMENT_VENDOR, REFERENCE_ID)
VALUES ('PENDING', TO_DATE('10/17/2022', 'MM/DD/YYYY'), 3, 'USPS', 1000000009);
INSERT INTO SHIPMENT ( SHIPMENT_STATUS, DATE_SHIPPED, TARGET_DELIVERY_DAYS,
SHIPMENT_VENDOR, REFERENCE_ID)
VALUES ('DELIVERED', TO_DATE('09/22/2022', 'MM/DD/YYYY'), 9, 'FEDEX', 1000000010);
INSERT INTO SHIPMENT ( SHIPMENT_STATUS, DATE_SHIPPED, TARGET_DELIVERY_DAYS,
SHIPMENT_VENDOR, REFERENCE_ID)
VALUES ('SHIPPED', TO_DATE('03/05/2022', 'MM/DD/YYYY'), 7, 'USPS', 1000000007);
INSERT INTO SHIPMENT ( SHIPMENT_STATUS, DATE_SHIPPED, TARGET_DELIVERY_DAYS,
SHIPMENT_VENDOR, REFERENCE_ID)
VALUES ('SHIPPED', TO_DATE('05/15/2022', 'MM/DD/YYYY'), 2, 'UPS', 1000000011);
INSERT INTO SHIPMENT ( SHIPMENT_STATUS, DATE_SHIPPED, TARGET_DELIVERY_DAYS,
SHIPMENT_VENDOR, REFERENCE_ID)
VALUES ('DELIVERED', TO_DATE('10/30/2022', 'MM/DD/YYYY'), 6, 'USPS', 1000000011);
INSERT INTO SHIPMENT ( SHIPMENT_STATUS, DATE_SHIPPED, TARGET_DELIVERY_DAYS,
SHIPMENT_VENDOR, REFERENCE_ID)
VALUES ('PENDING', TO_DATE('09/15/2022', 'MM/DD/YYYY'), 9, 'FEDEX', 1000000009);
INSERT INTO SHIPMENT ( SHIPMENT_STATUS, DATE_SHIPPED, TARGET_DELIVERY_DAYS,
SHIPMENT_VENDOR, REFERENCE_ID)
VALUES ('DELIVERED', TO_DATE('10/15/2022', 'MM/DD/YYYY'), 5, 'USPS', 1000000008);
INSERT INTO SHIPMENT ( SHIPMENT_STATUS, DATE_SHIPPED, TARGET_DELIVERY_DAYS,
SHIPMENT_VENDOR, REFERENCE_ID)
VALUES ('PENDING', TO_DATE('04/22/2022', 'MM/DD/YYYY'), 5, 'USPS', 1000000014);
```

```
INSERT INTO SHIPMENT ( SHIPMENT_STATUS, DATE_SHIPPED, TARGET_DELIVERY_DAYS,
SHIPMENT_VENDOR, REFERENCE_ID)
VALUES ('SHIPPED', TO_DATE('05/19/2022', 'MM/DD/YYYY'), 3, 'UPS', 1000000013);
INSERT INTO SHIPMENT ( SHIPMENT_STATUS, DATE_SHIPPED, TARGET_DELIVERY_DAYS,
SHIPMENT_VENDOR, REFERENCE_ID)
VALUES ('PENDING', TO_DATE('10/27/2022', 'MM/DD/YYYY'), 5, 'USPS', 1000000013);
INSERT INTO SHIPMENT ( SHIPMENT_STATUS, DATE_SHIPPED, TARGET_DELIVERY_DAYS,
SHIPMENT_VENDOR, REFERENCE_ID)
VALUES ('DELIVERED', TO_DATE('11/17/2022', 'MM/DD/YYYY'), 8, 'FEDEX', 1000000014);
INSERT INTO SHIPMENT ( SHIPMENT_STATUS, DATE_SHIPPED, TARGET_DELIVERY_DAYS,
SHIPMENT_VENDOR, REFERENCE_ID)
VALUES ('DELIVERED', TO_DATE('05/06/2022', 'MM/DD/YYYY'), 5, 'USPS', 1000000009);

-- PAYMENT_INFO
INSERT INTO PAYMENT_INFO ( MEMBERSHIP_NO, CARD_TYPE, CARD_NUMBER, CVV,
EXPIRY)
VALUES (1000000000, 'VISA', 4728372836284627, 123, TO_DATE('01/05/2022', 'DD/MM/YYYY'));
INSERT INTO PAYMENT_INFO ( MEMBERSHIP_NO, CARD_TYPE, CARD_NUMBER, CVV,
EXPIRY)
VALUES (1000000001, 'MASTERCARD', 6372836482937495, 456, TO_DATE('01/08/2023',
'DD/MM/YYYY'));
INSERT INTO PAYMENT_INFO ( MEMBERSHIP_NO, CARD_TYPE, CARD_NUMBER, CVV,
EXPIRY)
VALUES (1000000002, 'DISCOVER', 6347372168641557, 789, TO_DATE('01/10/2026',
'DD/MM/YYYY'));
INSERT INTO PAYMENT_INFO ( MEMBERSHIP_NO, CARD_TYPE, CARD_NUMBER, CVV,
EXPIRY)
VALUES (1000000003, 'VISA', 6583729816473846, 192, TO_DATE('01/11/2025', 'DD/MM/YYYY'));
INSERT INTO PAYMENT_INFO ( MEMBERSHIP_NO, CARD_TYPE, CARD_NUMBER, CVV,
EXPIRY)
VALUES (1000000004, 'MASTERCARD', 6483927562748374, 473, TO_DATE('01/04/2028',
'DD/MM/YYYY'));
INSERT INTO PAYMENT_INFO ( MEMBERSHIP_NO, CARD_TYPE, CARD_NUMBER, CVV,
EXPIRY)
```

```
VALUES (1000000005, 'VISA', 4638493746391001, 843, TO_DATE('01/08/2029', 'DD/MM/YYYY'));
INSERT INTO PAYMENT_INFO ( MEMBERSHIP_NO, CARD_TYPE, CARD_NUMBER, CVV,
EXPIRY)
VALUES (1000000005, 'DISCOVER', 6473846382251846, 482, TO_DATE('01/08/2025',
'DD/MM/YYYY'));
INSERT INTO PAYMENT_INFO ( MEMBERSHIP_NO, CARD_TYPE, CARD_NUMBER, CVV,
EXPIRY)
VALUES (1000000005, 'MASTERCARD', 4637592745283563, 894, TO_DATE('01/06/2027',
'DD/MM/YYYY'));
INSERT INTO PAYMENT_INFO ( MEMBERSHIP_NO, CARD_TYPE, CARD_NUMBER, CVV,
EXPIRY)
VALUES (1000000004, 'VISA', 6789421456774367, 950, TO_DATE('01/09/2029', 'DD/MM/YYYY'));
INSERT INTO PAYMENT_INFO ( MEMBERSHIP_NO, CARD_TYPE, CARD_NUMBER, CVV,
EXPIRY)
VALUES (1000000003, 'VISA', 6748365384957354, 923, TO_DATE('01/10/2024', 'DD/MM/YYYY'));
INSERT INTO PAYMENT_INFO ( MEMBERSHIP_NO, CARD_TYPE, CARD_NUMBER, CVV,
EXPIRY)
VALUES (1000000002, 'AMEX', 6378372836284627, 759, TO_DATE('01/11/2024', 'DD/MM/YYYY'));
INSERT INTO PAYMENT_INFO ( MEMBERSHIP_NO, CARD_TYPE, CARD_NUMBER, CVV,
EXPIRY)
VALUES (1000000001, 'VISA', 4728372836284699, 492, TO_DATE('01/08/2026', 'DD/MM/YYYY'));
INSERT INTO PAYMENT_INFO ( MEMBERSHIP_NO, CARD_TYPE, CARD_NUMBER, CVV,
EXPIRY)
VALUES (1000000003, 'MASTERCARD', 4728372628484627, 764, TO_DATE('01/08/2024',
'DD/MM/YYYY'));
INSERT INTO PAYMENT_INFO ( MEMBERSHIP_NO, CARD_TYPE, CARD_NUMBER, CVV,
EXPIRY)
VALUES (1000000004, 'AMEX', 2846389836284627, 942, TO_DATE('01/07/2025', 'DD/MM/YYYY'));
INSERT INTO PAYMENT_INFO ( MEMBERSHIP_NO, CARD_TYPE, CARD_NUMBER, CVV,
EXPIRY)
VALUES (1000000005, 'VISA', 4728372834628837, 763, TO_DATE('01/07/2024', 'DD/MM/YYYY'));

-- MANUFACTURER
```

```
INSERT INTO MANUFACTURER ( MANUFACTURER_NAME, STREET, CITY, COUNTRY,
EMAIL, PHONE)
VALUES ( 'Crown Corp.', '372 Haven Rd', 'Dallas', 'USA', 'crowncorp@gmail.com', '372-382-4629');
INSERT INTO MANUFACTURER ( MANUFACTURER_NAME, STREET, CITY, COUNTRY,
EMAIL, PHONE)
VALUES ( 'Sierra Industries', '472 McCain Dr', 'London', 'UK', 'sierra@gmail.com', '836-375-3755');
INSERT INTO MANUFACTURER ( MANUFACTURER_NAME, STREET, CITY, COUNTRY,
EMAIL, PHONE)
VALUES ( 'Pathways Co.', '376 Log Dr', 'San Jose', 'USA', 'pathways@gmail.com', '375-378-3758');
INSERT INTO MANUFACTURER ( MANUFACTURER_NAME, STREET, CITY, COUNTRY,
EMAIL, PHONE)
VALUES ( 'Singapore Retail', '375 District', 'Letz', 'Singapore', 'singretail@gmail.com', '375-375-2849');
INSERT INTO MANUFACTURER ( MANUFACTURER_NAME, STREET, CITY, COUNTRY,
EMAIL, PHONE)
VALUES ( 'Total Co', '3846 Main St', 'Dallas', 'USA', 'totalco@gmail.com', '214-265-3859');
INSERT INTO MANUFACTURER ( MANUFACTURER_NAME, STREET, CITY, COUNTRY,
EMAIL, PHONE)
VALUES ( 'ItaliaCo.', '476 Nice Rd', 'Bertz', 'Italy', 'italiaco@gmail.com', '375-274-2648');
INSERT INTO MANUFACTURER ( MANUFACTURER_NAME, STREET, CITY, COUNTRY,
EMAIL, PHONE)
VALUES ( 'ChinaRetails', '4678 District', 'Beijing', 'China', 'chinaretail@gmail.com', '375-274-2859');
INSERT INTO MANUFACTURER ( MANUFACTURER_NAME, STREET, CITY, COUNTRY,
EMAIL, PHONE)
VALUES ( 'RelianceCo.', '1919 Teltan Rd', 'Dallas', 'USA', 'relianceco@gmail.com', '332-462-9763');
INSERT INTO MANUFACTURER ( MANUFACTURER_NAME, STREET, CITY, COUNTRY,
EMAIL, PHONE)
VALUES ( 'ShapeMasters', '4768 Lakeway Dr', 'Houston', 'USA', 'shapemaster@gmail.com',
'275-284-2749');
INSERT INTO MANUFACTURER ( MANUFACTURER_NAME, STREET, CITY, COUNTRY,
EMAIL, PHONE)
VALUES ( 'Terrain Corp.', '376 Industry Ln', 'Chicago', 'USA', 'terrain@gmail.com', '376-274-2859');
INSERT INTO MANUFACTURER ( MANUFACTURER_NAME, STREET, CITY, COUNTRY,
EMAIL, PHONE)
VALUES ( 'TalentAgencies', '373 West Dr', 'Los Angeles', 'USA', 'talentag@gmail.com', '372-488-4619');
```

```
INSERT INTO MANUFACTURER ( MANUFACTURER_NAME, STREET, CITY, COUNTRY,
EMAIL, PHONE)
VALUES ( 'Templeton Co.', '3759 Mouse Ln', 'Saigon', 'Vietnam', 'templeton@gmail.com',
'375-388-1739');
INSERT INTO MANUFACTURER ( MANUFACTURER_NAME, STREET, CITY, COUNTRY,
EMAIL, PHONE)
VALUES ( 'PlatniumCo.', '759 Grad Dr', 'Dallas', 'USA', 'platco@gmail.com', '465-276-3725');
INSERT INTO MANUFACTURER ( MANUFACTURER_NAME, STREET, CITY, COUNTRY,
EMAIL, PHONE)
VALUES ( 'Commercial Co.', '3482 Mex Dr', 'Tijuana', 'Mexico', 'commercialco@gmail.com',
'476-276-2829');
INSERT INTO MANUFACTURER ( MANUFACTURER_NAME, STREET, CITY, COUNTRY,
EMAIL, PHONE)
VALUES ( 'America Industry', '376 Lake West', 'Dallas', 'USA', 'americaind@gmail.com',
'476-286-4628');

-- PRODUCT
INSERT INTO PRODUCT ( PRODUCT_NAME, PRICE, INVENTORY, TAXABLE, DISCOUNT,
MANUFACTURER_ID)
VALUES ( 'Rice', 4.50, 150, 1, 0.5, 100000);
INSERT INTO PRODUCT ( PRODUCT_NAME, PRICE, INVENTORY, TAXABLE, DISCOUNT,
MANUFACTURER_ID)
VALUES ( 'Shirt', 12.70, 280, 3, 1, 100001);
INSERT INTO PRODUCT ( PRODUCT_NAME, PRICE, INVENTORY, TAXABLE, DISCOUNT,
MANUFACTURER_ID)
VALUES ( 'Fruit', 3.50, 900, 1, 1, 100001);
INSERT INTO PRODUCT ( PRODUCT_NAME, PRICE, INVENTORY, TAXABLE, DISCOUNT,
MANUFACTURER_ID)
VALUES ( 'Decor', 32.60, 250, 4, 2.5, 100002);
INSERT INTO PRODUCT ( PRODUCT_NAME, PRICE, INVENTORY, TAXABLE, DISCOUNT,
MANUFACTURER_ID)
VALUES ( 'TV', 874.28, 36, 83.90, 20.45, 100003);
INSERT INTO PRODUCT ( PRODUCT_NAME, PRICE, INVENTORY, TAXABLE, DISCOUNT,
MANUFACTURER_ID)
```

```
VALUES ( 'Canned Food', 2.30, 1482, 0.75, 0.5, 100000);
INSERT INTO PRODUCT ( PRODUCT_NAME, PRICE, INVENTORY, TAXABLE, DISCOUNT,
MANUFACTURER_ID)
VALUES ( 'Dog Food', 29.50, 90, 2, 1.5, 100001);
INSERT INTO PRODUCT ( PRODUCT_NAME, PRICE, INVENTORY, TAXABLE, DISCOUNT,
MANUFACTURER_ID)
VALUES ( 'Corn', 1.50, 3000, 0.5, 0.2, 100005);
INSERT INTO PRODUCT ( PRODUCT_NAME, PRICE, INVENTORY, TAXABLE, DISCOUNT,
MANUFACTURER_ID)
VALUES ( 'Blender', 42.50, 190, 9, 2.2, 100008);
INSERT INTO PRODUCT ( PRODUCT_NAME, PRICE, INVENTORY, TAXABLE, DISCOUNT,
MANUFACTURER_ID)
VALUES ( 'Grill', 999.99, 110, 78.65, 5.5, 100009);
INSERT INTO PRODUCT ( PRODUCT_NAME, PRICE, INVENTORY, TAXABLE, DISCOUNT,
MANUFACTURER_ID)
VALUES ( 'Flower', 4.50, 1500, 1, 0.75, 100010);
INSERT INTO PRODUCT ( PRODUCT_NAME, PRICE, INVENTORY, TAXABLE, DISCOUNT,
MANUFACTURER_ID)
VALUES ( 'Bird Feed', 10.50, 60, 2, 1.5, 100007);
INSERT INTO PRODUCT ( PRODUCT_NAME, PRICE, INVENTORY, TAXABLE, DISCOUNT,
MANUFACTURER_ID)
VALUES ( 'Chicken', 4.50, 1550, 1, 0.5, 100012);
INSERT INTO PRODUCT ( PRODUCT_NAME, PRICE, INVENTORY, TAXABLE, DISCOUNT,
MANUFACTURER_ID)
VALUES ( 'Couch', 1444.50, 90, 98.7, 25.5, 100008);
INSERT INTO PRODUCT ( PRODUCT_NAME, PRICE, INVENTORY, TAXABLE, DISCOUNT,
MANUFACTURER_ID)
VALUES ( 'Shoes', 94.50, 180, 10.45, 4.5, 100003);



INSERT INTO ORDER_HAS_PRODUCTS (PRODUCT_ID, ORDER_ID)
VALUES (100013, 1000000001);
INSERT INTO ORDER_HAS_PRODUCTS (PRODUCT_ID, ORDER_ID)
```

```
VALUES (100012, 1000000002);
INSERT INTO ORDER_HAS_PRODUCTS (PRODUCT_ID, ORDER_ID)
VALUES (100011, 1000000003);
INSERT INTO ORDER_HAS_PRODUCTS (PRODUCT_ID, ORDER_ID)
VALUES (100014, 1000000004);
INSERT INTO ORDER_HAS_PRODUCTS (PRODUCT_ID, ORDER_ID)
VALUES (100010, 1000000005);
INSERT INTO ORDER_HAS_PRODUCTS (PRODUCT_ID, ORDER_ID)
VALUES (100009, 1000000006);
INSERT INTO ORDER_HAS_PRODUCTS (PRODUCT_ID, ORDER_ID)
VALUES (100008, 1000000007);
INSERT INTO ORDER_HAS_PRODUCTS (PRODUCT_ID, ORDER_ID)
VALUES (100007, 1000000008);
INSERT INTO ORDER_HAS_PRODUCTS (PRODUCT_ID, ORDER_ID)
VALUES (100006, 1000000009);
INSERT INTO ORDER_HAS_PRODUCTS (PRODUCT_ID, ORDER_ID)
VALUES (100005, 1000000010);
INSERT INTO ORDER_HAS_PRODUCTS (PRODUCT_ID, ORDER_ID)
VALUES (100004, 1000000011);
INSERT INTO ORDER_HAS_PRODUCTS (PRODUCT_ID, ORDER_ID)
VALUES (100003, 1000000012);
INSERT INTO ORDER_HAS_PRODUCTS (PRODUCT_ID, ORDER_ID)
VALUES (100002, 1000000013);
INSERT INTO ORDER_HAS_PRODUCTS (PRODUCT_ID, ORDER_ID)
VALUES (100001, 1000000013);
INSERT INTO ORDER_HAS_PRODUCTS (PRODUCT_ID, ORDER_ID)
VALUES (100000, 1000000010);

INSERT INTO STORE_SELL_PRODUCTS (STORE_ID, PRODUCT_ID)
VALUES (10000, 100000);
INSERT INTO STORE_SELL_PRODUCTS (STORE_ID, PRODUCT_ID)
VALUES (10002, 100001);
INSERT INTO STORE_SELL_PRODUCTS (STORE_ID, PRODUCT_ID)
VALUES (10005, 100013);
```

```
INSERT INTO STORE_SELL_PRODUCTS (STORE_ID, PRODUCT_ID)
VALUES (10002, 100014);
INSERT INTO STORE_SELL_PRODUCTS (STORE_ID, PRODUCT_ID)
VALUES (10003, 100005);
INSERT INTO STORE_SELL_PRODUCTS (STORE_ID, PRODUCT_ID)
VALUES (10004, 100004);
INSERT INTO STORE_SELL_PRODUCTS (STORE_ID, PRODUCT_ID)
VALUES (10005, 100006);
INSERT INTO STORE_SELL_PRODUCTS (STORE_ID, PRODUCT_ID)
VALUES (10006, 100003);
INSERT INTO STORE_SELL_PRODUCTS (STORE_ID, PRODUCT_ID)
VALUES (10004, 100002);
INSERT INTO STORE_SELL_PRODUCTS (STORE_ID, PRODUCT_ID)
VALUES (10003, 100007);
INSERT INTO STORE_SELL_PRODUCTS (STORE_ID, PRODUCT_ID)
VALUES (10002, 100008);
INSERT INTO STORE_SELL_PRODUCTS (STORE_ID, PRODUCT_ID)
VALUES (10002, 100009);
INSERT INTO STORE_SELL_PRODUCTS (STORE_ID, PRODUCT_ID)
VALUES (10000, 100010);
INSERT INTO STORE_SELL_PRODUCTS (STORE_ID, PRODUCT_ID)
VALUES (10005, 100011);
INSERT INTO STORE_SELL_PRODUCTS (STORE_ID, PRODUCT_ID)
VALUES (10004, 100012);


/* SQL QUERIES */
```

Q1. Select all columns and all rows from one table.

```
SELECT * FROM ALL_ORDER;
```

Q2. Select five columns and all rows from one table.

```
SELECT FIRST_NAME, LAST_NAME, DOB, EMAIL, PHONE
FROM CUSTOMER;
```

Q3. Select all columns from all rows from one view

```
SELECT * FROM PRODUCT;
```

Q4. Using a join on 2 tables, select all columns and all rows.

```
SELECT *
FROM MANUFACTURER A
LEFT JOIN PRODUCT B
ON A.MANUFACTURER_ID = B.MANUFACTURER_ID;
```

Q5. Select and order data retrieved from one table.

```
SELECT * FROM PRODUCT
ORDER BY PRICE DESC;
```

Q6. Using a join on 3 tables, select 5 columns from the 3 tables. Use syntax that would limit the output to 10 rows.

```
SELECT A.ORDER_ID, C.PRODUCT_ID, C.PRODUCT_NAME
FROM ALL_ORDER A
JOIN ORDER_HAS_PRODUCTS B ON A.ORDER_ID = B.ORDER_ID
JOIN PRODUCT C ON B.PRODUCT_ID = C.PRODUCT_ID
FETCH FIRST 10 ROWS ONLY;
```

Q7. Select distinct rows using joins on 3 tables.

```
SELECT DISTINCT *
FROM ALL_ORDER A
JOIN ORDER_HAS_PRODUCTS B ON A.ORDER_ID = B.ORDER_ID
JOIN PRODUCT C ON B.PRODUCT_ID = C.PRODUCT_ID;
```

Q8. Use GROUP BY and HAVING in a select statement using one or more tables.

```
SELECT B.SHIPMENT_STATUS, SUM(TOTAL_AMOUNT) AS T_AMOUNT

FROM ALL_ORDER A

JOIN SHIPMENT B ON A.REFERENCE_ID=B.REFERENCE_ID

GROUP BY B.SHIPMENT_STATUS

WHERE TOTAL_AMOUNT>100;
```

Q9. Use IN clause to select data from one or more tables.

```
SELECT * FROM EMPLOYEE

WHERE ZIPCODE IN (75252,75080,75010);
```

Q10. Select length of one column from one table.

```
SELECT LENGTH(PRODUCT_NAME) FROM PRODUCT;
```

Q11. Delete one record from one table. Use select statements to demonstrate the table contents before and after the DELETE statement. Make sure you use ROLLBACKafterwards so that the data will not be physically removed.

```
SELECT * FROM EMPLOYEE;

DELETE FROM EMPLOYEE WHERE EMPLOYEE_ID = 1000000;

SELECT * FROM EMPLOYEE;

ROLLBACK;
```

Q12. Update one record from one table. Use select statements to demonstrate the table contents before and after the UPDATE statement. Make sure you use ROLLBACK afterwards so that the data will not be physically removed.

```
SELECT * FROM EMPLOYEE;

UPDATE EMPLOYEE SET City = 'Austin' WHERE EMPLOYEE_ID = 1000002;

Select * from EMPLOYEE;

ROLLBACK;
```

Q13. Count no of online orders.

```
SELECT COUNT(DISTINCT A.ORDER_ID) AS ONLINE_ORDERS

FROM ALL_ORDER A

JOIN ONLINE_ORDER B

ON A.ORDER_ID = B.ORDER_ID;
```

Q14. Tell the average number of products per order.

```
SELECT ORDER_ID, AVG(TOTAL_AMOUNT) AS AVG_AMOUNT

FROM ALL_ORDER

GROUP BY ORDER_ID;
```

Q15. LIST CUSTOMERS WITH STATUS.

```
SELECT A.CUSTOMER_ID, B.STATUS

FROM CUSTOMER A

JOIN ALL_ORDER B ON A.CUSTOMER_ID=B.CUSTOMER_ID;
```

Q16. List number of orders grouped by shipped status.

```
SELECT SHIPMENT_STATUS, COUNT(DISTINCT ORDER_ID) AS orders

FROM SHIPMENT A

JOIN ONLINE_ORDER B ON A.ORDER_ID=B.ORDER_ID

JOIN ALL_ORDER C ON B.Order_ID=C.ORDER_ID

GROUP BY 1;
```

Q17. Tell the average number of payments done by customers who are also members.

```
SELECT CUSTOMER_ID, AVG(PAYMENT_ID) AS AVG_PAYMENTS

FROM CUSTOMER A

JOIN MEMBER B ON A.CUSTOMER_ID=B.CUSTOMER_ID

JOIN PAYMENT_INFO C ON B.MEMBERSHIP_NO=C.MEMBERSHIP_NO
```

```
GROUP BY 1;
```

Q18. List the number of stores and employees grouped by city.

```
SELECT A.CITY, COUNT(DISTINCT STORE_ID) AS STORES, COUNT(DISTINCT EMPLOYEE_ID) AS EMPLOYEES
FROM STORE_INFO A
LEFT JOIN EMPLOYEE B on A.CITY=B.CITY
GROUP BY 1;
```

Q19. List the orders and their target delivery days.

```
SELECT A.ORDER_ID, C.TARGET_DELIVERY_DAYS
FROM ALL_ORDER A
JOIN ONLINE_ORDER B
ON A.ORDER_ID=B.ORDER_ID
JOIN SHIPMENT C ON B.ORDER_ID=C.ORDER_ID;
```

Q20. List the manufacturers and maximum discount offered by them on all products.

```
SELECT A.MANUFACTURER_ID, A.MANUFACTURER_NAME, MAX(B.DISCOUNT) AS MAX_DISCOUNT
FROM MANUFACTURER A
JOIN PRODUCT B
ON A.MANUFACTURER_ID = B.MANUFACTURER_ID
GROUP BY 1,2;
```

DDL, DML, and Query Output DDL Output.

DDL

Number	Elapsed	Statement	Feedback	Rows
1	0.05	DROP SEQUENCE SEQ_STORE_ID	Sequence dropped.	0

2	0.01	DROP SEQUENCE SEQ_PRODUCT_ID	Sequence dropped.	0
3	0.01	DROP SEQUENCE SEQ_EMPLOYEE_ID	Sequence dropped.	0
4	0	DROP SEQUENCE SEQ_MANUFACTURER_ID	Sequence dropped.	0
5	0	DROP SEQUENCE SEQ_MEMBERSHIP_NO	Sequence dropped.	0
6	0.01	DROP SEQUENCE SEQ_ORDER_ID	Sequence dropped.	0
7	0.01	DROP SEQUENCE SEQ_REFERENCE_ID	Sequence dropped.	0
8	0.01	DROP SEQUENCE SEQ_PAYMENT_ID	Sequence dropped.	0
9	0.01	DROP SEQUENCE SEQ_SHIPMENT_ID	Sequence dropped.	0
10	0.06	DROP INDEX IDX_EMPLOYEE_ZIPCODE	Index dropped.	0
11	0.07	DROP INDEX IDX_MANUFACTURER_COUNTRY	Index dropped.	0
12	0.03	DROP INDEX IDX_PRODUCT_MANUFACTURER	Index dropped.	0
13	0.04	DROP INDEX IDX_PRODUCT_PRICE	Index dropped.	0
14	0.04	DROP INDEX IDX_PRODUCT_INVENTORY	Index dropped.	0
15	0.04	DROP INDEX IDX_PRODUCT_TAXABLE	Index dropped.	0
16	0.03	DROP INDEX IDX_CUSTOMER_PHONE	Index dropped.	0
17	0.03	DROP INDEX IDX_CUSTOMER_ZIPCODE	Index dropped.	0
18	0.06	DROP INDEX IDX_CUSTOMER_DOB	Index dropped.	0
19	0.06	DROP INDEX IDX_CUSTOMER_STATE	Index dropped.	0
20	0.06	DROP INDEX IDX_PAYMENT_CARD_TYPE	Index dropped.	0
21	0.06	DROP INDEX IDX_ALL_ORDER_USER_ID	Index dropped.	0
22	0.04	DROP INDEX IDX_ALL_ORDER_STATUS	Index dropped.	0
23	0.04	DROP INDEX IDX_ALL_ORDER_DATE	Index dropped.	0
24	0.04	DROP INDEX IDX_ALL_ORDER_TOTAL_AMOUNT	Index dropped.	0
25	0.03	DROP INDEX IDX_SHIPMENT_REFERENCE_ID	Index dropped.	0
26	0.04	DROP INDEX IDX_SHIPMENT_STATUS	Index dropped.	0
27	0.04	DROP INDEX IDX_SHIPMENT_TARGET_DELIVERY	Index dropped.	0

28	**0.07**	**DROP INDEX IDX_SHIPMENT_VENDOR**	**Index dropped.**	**0**
29	**0.61**	**DROP TABLE ORDER_HAS_PRODUCTS**	**Table dropped.**	**0**
30	**0.56**	**DROP TABLE STORE_SELL_PRODUCTS**	**Table dropped.**	**0**
31	**0.64**	**DROP TABLE PRODUCT**	**Table dropped.**	**0**
32	**0.62**	**DROP TABLE MANUFACTURER**	**Table dropped.**	**0**
33	**0.64**	**DROP TABLE EMPLOYEE**	**Table dropped.**	**0**
34	**0.65**	**DROP TABLE PAYMENT_INFO**	**Table dropped.**	**0**
35	**0.62**	**DROP TABLE SHIPMENT**	**Table dropped.**	**0**
36	**0.67**	**DROP TABLE ALL_ORDER**	**Table dropped.**	**0**
37	**0.67**	**DROP TABLE CUSTOMER**	**Table dropped.**	**0**
38	**0.61**	**DROP TABLE STORE_INFO**	**Table dropped.**	**0**
39	**0.16**	**CREATE TABLE STORE_INFO(STORE_ID INTEGER**	**Table created.**	**0**
40	**0.07**	**CREATE TABLE EMPLOYEE(EMPLOYEE_ID INTEGER**	**Table created.**	**0**
41	**0.03**	**CREATE TABLE MANUFACTURER(MANUFACTURER_ID INTEGER**	**Table created.**	**0**
42	**0.03**	**CREATE TABLE PRODUCT(PRODUCT_ID INTEGER**	**Table created.**	**0**
43	**0.05**	**CREATE TABLE CUSTOMER(CUSTOMER_ID VARCHAR2(30)**	**Table created.**	**0**
44	**0.03**	**CREATE TABLE PAYMENT_INFO(PAYMENT_ID INTEGER**	**Table created.**	**0**
45	**0.05**	**CREATE TABLE ALL_ORDER(ORDER_ID INTEGER**	**Table created.**	**0**
46	**0.03**	**CREATE TABLE SHIPMENT(SHIPMENT_ID INTEGER**	**Table created.**	**0**
47	**0.03**	**CREATE TABLE ORDER_HAS_PRODUCTS(ORDER_ID INT**	**Table created.**	**0**
48	**0.03**	**CREATE TABLE STORE_SELL_PRODUCTS(STORE_ID IN**	**Table created.**	**0**
49	**0.02**	**CREATE INDEX IDX_EMPLOYEE_ZIPCODE ON EMPLOYEE (ZIPCODE)**	**Index created.**	**0**
50	**0.01**	**CREATE INDEX IDX_MANUFACTURER_COUNTRY ON MANUFACTURER (COUNT**	**Index created.**	**0**
51	**0.01**	**CREATE INDEX IDX_PRODUCT_MANUFACTURER ON PRODUCT (MANUFACTUR**	**Index created.**	**0**
52	**0.01**	**CREATE INDEX IDX_PRODUCT_PRICE ON PRODUCT (PRICE)**	**Index created.**	**0**
53	**0.01**	**CREATE INDEX IDX_PRODUCT_INVENTORY ON PRODUCT (INVENTORY)**	**Index created.**	**0**
54	**0.01**	**CREATE INDEX IDX_PRODUCT_TAXABLE ON PRODUCT (TAXABLE)**	**Index created.**	**0**

55	0.01	CREATE INDEX IDX_CUSTOMER_PHONE ON CUSTOMER (PHONE)	Index created.	0
56	0.01	CREATE INDEX IDX_CUSTOMER_ZIPCODE ON CUSTOMER (ZIPCODE)	Index created.	0
57	0.01	CREATE INDEX IDX_CUSTOMER_DOB ON CUSTOMER (DOB)	Index created.	0
58	0.01	CREATE INDEX IDX_CUSTOMER_STATE ON CUSTOMER (STATE_LOC)	Index created.	0
59	0.01	CREATE INDEX IDX_IF_MEMBER ON CUSTOMER (IF_MEMBER)	Index created.	0
60	0.01	CREATE INDEX IDX_PAYMENT_CARD_TYPE ON PAYMENT_INFO (CARD_TYP	Index created.	0
61	0.01	CREATE INDEX IDX_ALL_ORDER_USER_ID ON ALL_ORDER(CUSTOMER_ID)	Index created.	0
62	0.01	CREATE INDEX IDX_ALL_ORDER_STATUS ON ALL_ORDER(STATUS)	Index created.	0
63	0.01	CREATE INDEX IDX_ALL_ORDER_DATE ON ALL_ORDER(ORDER_DATE)	Index created.	0
64	0.01	CREATE INDEX IDX_ALL_ORDER_ITEMS ON ALL_ORDER(NO_ITEMS)	Index created.	0
65	0.01	CREATE INDEX IDX_ALL_ORDER_TOTAL_AMOUNT ON ALL_ORDER(TOTAL_A	Index created.	0
66	0.01	CREATE INDEX IDX_SHIPMENT_REFERENCE_ID ON SHIPMENT (REFERENC	Index created.	0
67	0.01	CREATE INDEX IDX_SHIPMENT_STATUS ON SHIPMENT (SHIPMENT_STATU	Index created.	0
68	0.01	CREATE INDEX IDX_SHIPMENT_TARGET_DELIVERY ON SHIPMENT (TARGE	Index created.	0
69	0.01	CREATE INDEX IDX_SHIPMENT_VENDOR ON SHIPMENT (SHIPMENT_VENDO	Index created.	0
70	0.07	ALTER TABLE STORE_INFO ADD (CREATED_BY VARCHAR2(30), DATE_C	Table altered.	0
71	0.06	ALTER TABLE EMPLOYEE ADD (CREATED_BY VARCHAR2(30), DATE_CRE	Table altered.	0
72	0.06	ALTER TABLE MANUFACTURER ADD (CREATED_BY VARCHAR2(30), DATE	Table altered.	0
73	0.06	ALTER TABLE CUSTOMER ADD (CREATED_BY VARCHAR2(30), DATE_CRE	Table altered.	0
74	0.06	ALTER TABLE ALL_ORDER ADD (CREATED_BY VARCHAR2(30), DATE_CR	Table altered.	0
75	0.07	ALTER TABLE PAYMENT_INFO ADD (CREATED_BY VARCHAR2(30), DATE	Table altered.	0
76	0.06	ALTER TABLE PRODUCT ADD (CREATED_BY VARCHAR2(30), DATE_CREA	Table altered.	0
77	0.06	ALTER TABLE SHIPMENT ADD (CREATED_BY VARCHAR2(30), DATE_CRE	Table altered.	0
78	0.08	CREATE OR REPLACE VIEW MEMBERS AS SELECT CUSTOMER_ID, IF_MEM	View created.	0
79	0.05	CREATE OR REPLACE VIEW NON_MEMBER_CUSTOMERS AS SELECT CUSTOM	View created.	0
80	0.09	CREATE OR REPLACE VIEW OFFLINE_ORDER AS SELECT ORDER_ID, IF	View created.	0

81	0.05	CREATE OR REPLACE VIEW ONLINE_ORDER AS SELECT ORDER_ID, IF_	View created.	0
82	0.12	CREATE SEQUENCE SEQ_STORE_ID INCREMENT BY 1 START WI	Sequence created.	0
83	0.01	CREATE SEQUENCE SEQ_EMPLOYEE_ID INCREMENT BY 1 START	Sequence created.	0
84	0.01	CREATE SEQUENCE SEQ_MANUFACTURER_ID INCREMENT BY 1 S	Sequence created.	0
85	0	CREATE SEQUENCE SEQ_PRODUCT_ID INCREMENT BY 1 START	Sequence created.	0
86	0.01	CREATE SEQUENCE SEQ_MEMBERSHIP_NO INCREMENT BY 1 STA	Sequence created.	0
87	0.01	CREATE SEQUENCE SEQ_ORDER_ID INCREMENT BY 1 START WI	Sequence created.	0
88	0	CREATE SEQUENCE SEQ_REFERENCE_ID INCREMENT BY 1 STAR	Sequence created.	0
89	0.01	CREATE SEQUENCE SEQ_PAYMENT_ID INCREMENT BY 1 START	Sequence created.	0
90	0	CREATE SEQUENCE SEQ_SHIPMENT_ID INCREMENT BY 1 START	Sequence created.	0

Number	Elapsed	Statement	Feedback	Rows
1	0.26	CREATE OR REPLACE TRIGGER TRG_STORE BEFORE INSERT OR UPD	Trigger created.	0
2	0.13	CREATE OR REPLACE TRIGGER TRG_EMPLOYEE BEFORE INSERT OR	Trigger created.	0
3	0.17	CREATE OR REPLACE TRIGGER TRG_MANUFACTURER BEFORE INSERT	Trigger created.	0
4	0.12	CREATE OR REPLACE TRIGGER TRG_PRODUCT BEFORE INSERT OR U	Trigger created.	0
5	0.17	CREATE OR REPLACE TRIGGER TRG_CUSTOMER BEFORE INSERT OR	Trigger created.	0

6	0.16	CREATE OR REPLACE TRIGGER TRG_ORDER BEFORE INSERT OR UPD	Trigger created.	0
7	0.16	CREATE OR REPLACE TRIGGER TRG_PAYMENT BEFORE INSERT OR U	Trigger created.	0
8	0.13	CREATE OR REPLACE TRIGGER TRG_SHIPMENT BEFORE INSERT OR	Trigger created.	0

DML

Number	Elapsed	Statement	Feedback	Rows
1	0.12	INSERT INTO STORE_INFO (STORE_LOC, STREET, CITY, STATE_NAME	1 row(s) inserted.	1
2	0	INSERT INTO STORE_INFO (STORE_LOC, STREET, CITY, STATE_NAME	1 row(s) inserted.	1
3	0.01	INSERT INTO STORE_INFO (STORE_LOC, STREET, CITY, STATE_NAME,	1 row(s) inserted.	1
4	0	INSERT INTO STORE_INFO (STORE_LOC, STREET, CITY, STATE_NAME	1 row(s) inserted.	1
5	0.01	INSERT INTO STORE_INFO (STORE_LOC, STREET, CITY, STATE_NAME	1 row(s) inserted.	1
6	0	INSERT INTO STORE_INFO (STORE_LOC, STREET, CITY, STATE_NAME	1 row(s) inserted.	1
7	0.01	INSERT INTO STORE_INFO (STORE_LOC, STREET, CITY, STATE_NAME	1 row(s) inserted.	1
8	0	INSERT INTO STORE_INFO (STORE_LOC, STREET, CITY, STATE_NAME	1 row(s) inserted.	1
9	0	INSERT INTO STORE_INFO (STORE_LOC, STREET, CITY,	1 row(s)	1

		STATE_NAME	inserted.	
10	0.02	INSERT INTO STORE_INFO (STORE_LOC, STREET, CITY, STATE_NAME	1 row(s) inserted.	1
11	0	INSERT INTO STORE_INFO (STORE_LOC, STREET, CITY, STATE_NAME	1 row(s) inserted.	1
12	0.01	INSERT INTO STORE_INFO (STORE_LOC, STREET, CITY, STATE_NAME	1 row(s) inserted.	1
13	0	INSERT INTO STORE_INFO (STORE_LOC, STREET, CITY, STATE_NAME	1 row(s) inserted.	1
14	0	INSERT INTO STORE_INFO (STORE_LOC, STREET, CITY, STATE_NAME	1 row(s) inserted.	1
15	0	INSERT INTO STORE_INFO (STORE_LOC, STREET, CITY, STATE_NAME	1 row(s) inserted.	1
16	0.04	INSERT INTO EMPLOYEE (STORE_ID, FIRST_NAME, MIDDLE, LAST_NA	1 row(s) inserted.	1
17	0	INSERT INTO EMPLOYEE (STORE_ID, FIRST_NAME, MIDDLE, LAST_NA	1 row(s) inserted.	1
18	0	INSERT INTO EMPLOYEE (STORE_ID, FIRST_NAME, MIDDLE, LAST_NA	1 row(s) inserted.	1
19	0	INSERT INTO EMPLOYEE (STORE_ID, FIRST_NAME, MIDDLE, LAST_NA	1 row(s) inserted.	1
20	0.01	INSERT INTO EMPLOYEE (STORE_ID, FIRST_NAME, MIDDLE, LAST_NA	1 row(s) inserted.	1
21	0	INSERT INTO EMPLOYEE (STORE_ID, FIRST_NAME, MIDDLE, LAST_NA	1 row(s) inserted.	1
22	0.01	INSERT INTO EMPLOYEE (STORE_ID, FIRST_NAME, MIDDLE, LAST_NA	1 row(s) inserted.	1

23	0	INSERT INTO EMPLOYEE (STORE_ID, FIRST_NAME, MIDDLE, LAST_NA	1 row(s) inserted.	1
24	0	INSERT INTO EMPLOYEE (STORE_ID, FIRST_NAME, MIDDLE, LAST_NA	1 row(s) inserted.	1
25	0	INSERT INTO EMPLOYEE (STORE_ID, FIRST_NAME, MIDDLE, LAST_NA	1 row(s) inserted.	1
26	0	INSERT INTO EMPLOYEE (STORE_ID, FIRST_NAME, MIDDLE, LAST_NA	1 row(s) inserted.	1
27	0.01	INSERT INTO EMPLOYEE (STORE_ID, FIRST_NAME, MIDDLE, LAST_NA	1 row(s) inserted.	1
28	0	INSERT INTO EMPLOYEE (STORE_ID, FIRST_NAME, MIDDLE, LAST_NA	1 row(s) inserted.	1
29	0.01	INSERT INTO EMPLOYEE (STORE_ID, FIRST_NAME, MIDDLE, LAST_NA	1 row(s) inserted.	1
30	0	INSERT INTO EMPLOYEE (STORE_ID, FIRST_NAME, MIDDLE, LAST_NA	1 row(s) inserted.	1
31	0.08	INSERT INTO CUSTOMER (CUSTOMER_ID, IF_MEMBER, FIRST_NAME, MI	1 row(s) inserted.	1
32	0	INSERT INTO CUSTOMER (CUSTOMER_ID, IF_MEMBER, FIRST_NAME, MI	1 row(s) inserted.	1
33	0	INSERT INTO CUSTOMER (CUSTOMER_ID, IF_MEMBER, FIRST_NAME, MI	1 row(s) inserted.	1
34	0	INSERT INTO CUSTOMER (CUSTOMER_ID, IF_MEMBER, FIRST_NAME, MI	1 row(s) inserted.	1
35	0	INSERT INTO CUSTOMER (CUSTOMER_ID, IF_MEMBER, FIRST_NAME, MI	1 row(s) inserted.	1
36	0.01	INSERT INTO CUSTOMER (CUSTOMER_ID, IF_MEMBER, FIRST_NAME, MI	1 row(s) inserted.	1

37	0	INSERT INTO CUSTOMER (CUSTOMER_ID, IF_MEMBER, FIRST_NAME, MI	1 row(s) inserted.	1
38	0	INSERT INTO CUSTOMER (CUSTOMER_ID, IF_MEMBER, FIRST_NAME, MI	1 row(s) inserted.	1
39	0	INSERT INTO CUSTOMER (CUSTOMER_ID, IF_MEMBER, FIRST_NAME, MI	1 row(s) inserted.	1
40	0	INSERT INTO CUSTOMER (CUSTOMER_ID, IF_MEMBER, FIRST_NAME, MI	1 row(s) inserted.	1
41	0.01	INSERT INTO CUSTOMER (CUSTOMER_ID, IF_MEMBER, FIRST_NAME, MI	1 row(s) inserted.	1
42	0	INSERT INTO CUSTOMER (CUSTOMER_ID, IF_MEMBER, FIRST_NAME, MI	1 row(s) inserted.	1
43	0	INSERT INTO CUSTOMER (CUSTOMER_ID, IF_MEMBER, FIRST_NAME, MI	1 row(s) inserted.	1
44	0.01	INSERT INTO CUSTOMER (CUSTOMER_ID, IF_MEMBER, FIRST_NAME, MI	1 row(s) inserted.	1
45	0	INSERT INTO CUSTOMER (CUSTOMER_ID, IF_MEMBER, FIRST_NAME, MI	1 row(s) inserted.	1
46	0.08	INSERT INTO ALL_ORDER (IF_ONLINE, STATUS, ORDER_DATE, NO_IT	1 row(s) inserted.	1
47	0	INSERT INTO ALL_ORDER (IF_ONLINE, STATUS, ORDER_DATE, NO_IT	1 row(s) inserted.	1
48	0	INSERT INTO ALL_ORDER (IF_ONLINE, STATUS, ORDER_DATE, NO_IT	1 row(s) inserted.	1
49	0	INSERT INTO ALL_ORDER (IF_ONLINE, STATUS, ORDER_DATE, NO_IT	1 row(s) inserted.	1
50	0.01	INSERT INTO ALL_ORDER (IF_ONLINE, STATUS, ORDER_DATE, NO_IT	1 row(s) inserted.	1

51	0.01	INSERT INTO ALL_ORDER (IF_ONLINE, STATUS, ORDER_DATE, NO_IT	1 row(s) inserted.	1
52	0	INSERT INTO ALL_ORDER (IF_ONLINE, STATUS, ORDER_DATE, NO_IT	1 row(s) inserted.	1
53	0	INSERT INTO ALL_ORDER (IF_ONLINE, STATUS, ORDER_DATE, NO_IT	1 row(s) inserted.	1
54	0	INSERT INTO ALL_ORDER (IF_ONLINE, STATUS, ORDER_DATE, NO_IT	1 row(s) inserted.	1
55	0	INSERT INTO ALL_ORDER (IF_ONLINE, STATUS, ORDER_DATE, NO_IT	1 row(s) inserted.	1
56	0.01	INSERT INTO ALL_ORDER (IF_ONLINE, STATUS, ORDER_DATE, NO_IT	1 row(s) inserted.	1
57	0	INSERT INTO ALL_ORDER (IF_ONLINE, STATUS, ORDER_DATE, NO_IT	1 row(s) inserted.	1
58	0.01	INSERT INTO ALL_ORDER (IF_ONLINE, STATUS, ORDER_DATE, NO_IT	1 row(s) inserted.	1
59	0	INSERT INTO ALL_ORDER (IF_ONLINE, STATUS, ORDER_DATE, NO_IT	1 row(s) inserted.	1
60	0	INSERT INTO ALL_ORDER (IF_ONLINE, STATUS, ORDER_DATE, NO_IT	1 row(s) inserted.	1
61	0.07	INSERT INTO SHIPMENT (SHIPMENT_STATUS, DATE_SHIPPED, TARGET	1 row(s) inserted.	1
62	0	INSERT INTO SHIPMENT (SHIPMENT_STATUS, DATE_SHIPPED, TARGET	1 row(s) inserted.	1
63	0	INSERT INTO SHIPMENT (SHIPMENT_STATUS, DATE_SHIPPED, TARGET	1 row(s) inserted.	1
64	0	INSERT INTO SHIPMENT (SHIPMENT_STATUS, DATE_SHIPPED, TARGET	1 row(s) inserted.	1

65	0.01	INSERT INTO SHIPMENT (SHIPMENT_STATUS, DATE_SHIPPED, TARGET	1 row(s) inserted.	1
66	0	INSERT INTO SHIPMENT (SHIPMENT_STATUS, DATE_SHIPPED, TARGET	1 row(s) inserted.	1
67	0	INSERT INTO SHIPMENT (SHIPMENT_STATUS, DATE_SHIPPED, TARGET	1 row(s) inserted.	1
68	0.02	INSERT INTO SHIPMENT (SHIPMENT_STATUS, DATE_SHIPPED, TARGET	1 row(s) inserted.	1
69	0	INSERT INTO SHIPMENT (SHIPMENT_STATUS, DATE_SHIPPED, TARGET	1 row(s) inserted.	1
70	0.01	INSERT INTO SHIPMENT (SHIPMENT_STATUS, DATE_SHIPPED, TARGET	1 row(s) inserted.	1
71	0	INSERT INTO SHIPMENT (SHIPMENT_STATUS, DATE_SHIPPED, TARGET	1 row(s) inserted.	1
72	0	INSERT INTO SHIPMENT (SHIPMENT_STATUS, DATE_SHIPPED, TARGET	1 row(s) inserted.	1
73	0.01	INSERT INTO SHIPMENT (SHIPMENT_STATUS, DATE_SHIPPED, TARGET	1 row(s) inserted.	1
74	0	INSERT INTO SHIPMENT (SHIPMENT_STATUS, DATE_SHIPPED, TARGET	1 row(s) inserted.	1
75	0	INSERT INTO SHIPMENT (SHIPMENT_STATUS, DATE_SHIPPED, TARGET	1 row(s) inserted.	1
76	0.03	INSERT INTO PAYMENT_INFO (MEMBERSHIP_NO, CARD_TYPE, CARD_NU	1 row(s) inserted.	1
77	0	INSERT INTO PAYMENT_INFO (MEMBERSHIP_NO, CARD_TYPE, CARD_NU	1 row(s) inserted.	1
78	0.01	INSERT INTO PAYMENT_INFO (MEMBERSHIP_NO, CARD_TYPE, CARD_NU	1 row(s) inserted.	1

79	0	INSERT INTO PAYMENT_INFO (MEMBERSHIP_NO, CARD_TYPE, CARD_NU	1 row(s) inserted.	1
80	0	INSERT INTO PAYMENT_INFO (MEMBERSHIP_NO, CARD_TYPE, CARD_NU	1 row(s) inserted.	1
81	0.01	INSERT INTO PAYMENT_INFO (MEMBERSHIP_NO, CARD_TYPE, CARD_NU	1 row(s) inserted.	1
82	0	INSERT INTO PAYMENT_INFO (MEMBERSHIP_NO, CARD_TYPE, CARD_NU	1 row(s) inserted.	1
83	0	INSERT INTO PAYMENT_INFO (MEMBERSHIP_NO, CARD_TYPE, CARD_NU	1 row(s) inserted.	1
84	0.01	INSERT INTO PAYMENT_INFO (MEMBERSHIP_NO, CARD_TYPE, CARD_NU	1 row(s) inserted.	1
85	0	INSERT INTO PAYMENT_INFO (MEMBERSHIP_NO, CARD_TYPE, CARD_NU	1 row(s) inserted.	1
86	0	INSERT INTO PAYMENT_INFO (MEMBERSHIP_NO, CARD_TYPE, CARD_NU	1 row(s) inserted.	1
87	0	INSERT INTO PAYMENT_INFO (MEMBERSHIP_NO, CARD_TYPE, CARD_NU	1 row(s) inserted.	1
88	0	INSERT INTO PAYMENT_INFO (MEMBERSHIP_NO, CARD_TYPE, CARD_NU	1 row(s) inserted.	1
89	0	INSERT INTO PAYMENT_INFO (MEMBERSHIP_NO, CARD_TYPE, CARD_NU	1 row(s) inserted.	1
90	0	INSERT INTO PAYMENT_INFO (MEMBERSHIP_NO, CARD_TYPE, CARD_NU	1 row(s) inserted.	1
91	0.02	INSERT INTO MANUFACTURER (MANUFACTURER_NAME, STREET, CITY,	1 row(s) inserted.	1
92	0.01	INSERT INTO MANUFACTURER (MANUFACTURER_NAME, STREET, CITY,	1 row(s) inserted.	1

93	0	INSERT INTO MANUFACTURER (MANUFACTURER_NAME, STREET, CITY,	1 row(s) inserted.	1
94	0.01	INSERT INTO MANUFACTURER (MANUFACTURER_NAME, STREET, CITY,	1 row(s) inserted.	1
95	0	INSERT INTO MANUFACTURER (MANUFACTURER_NAME, STREET, CITY,	1 row(s) inserted.	1
96	0	INSERT INTO MANUFACTURER (MANUFACTURER_NAME, STREET, CITY,	1 row(s) inserted.	1
97	0.01	INSERT INTO MANUFACTURER (MANUFACTURER_NAME, STREET, CITY,	1 row(s) inserted.	1
98	0	INSERT INTO MANUFACTURER (MANUFACTURER_NAME, STREET, CITY,	1 row(s) inserted.	1
99	0	INSERT INTO MANUFACTURER (MANUFACTURER_NAME, STREET, CITY,	1 row(s) inserted.	1
100	0.01	INSERT INTO MANUFACTURER (MANUFACTURER_NAME, STREET, CITY,	1 row(s) inserted.	1
101	0	INSERT INTO MANUFACTURER (MANUFACTURER_NAME, STREET, CITY,	1 row(s) inserted.	1
102	0	INSERT INTO MANUFACTURER (MANUFACTURER_NAME, STREET, CITY,	1 row(s) inserted.	1
103	0.01	INSERT INTO MANUFACTURER (MANUFACTURER_NAME, STREET, CITY,	1 row(s) inserted.	1
104	0	INSERT INTO MANUFACTURER (MANUFACTURER_NAME, STREET, CITY,	1 row(s) inserted.	1
105	0	INSERT INTO MANUFACTURER (MANUFACTURER_NAME, STREET, CITY,	1 row(s) inserted.	1
106	0.04	INSERT INTO PRODUCT (PRODUCT_NAME, PRICE, INVENTORY, TAXABL	1 row(s) inserted.	1

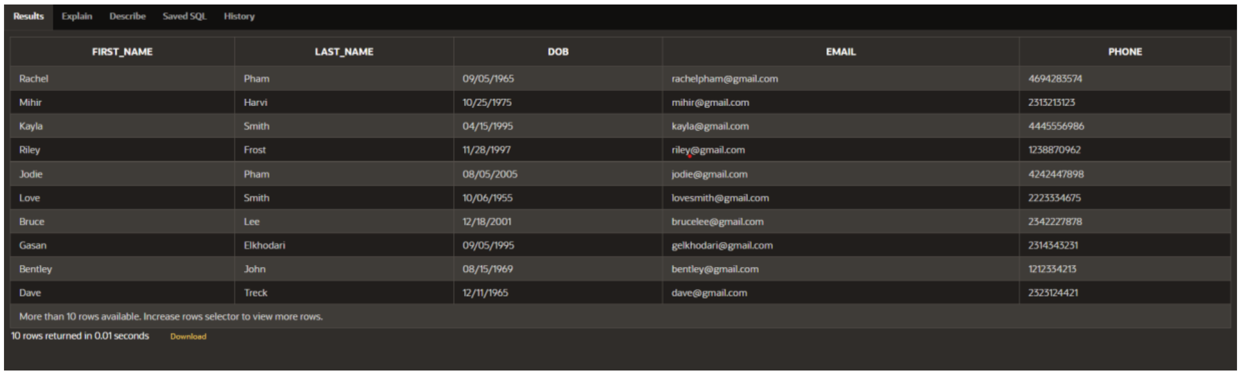
107	0.01	INSERT INTO PRODUCT (PRODUCT_NAME, PRICE, INVENTORY, TAXABL	1 row(s) inserted.	1
108	0	INSERT INTO PRODUCT (PRODUCT_NAME, PRICE, INVENTORY, TAXABL	1 row(s) inserted.	1
109	0	INSERT INTO PRODUCT (PRODUCT_NAME, PRICE, INVENTORY, TAXABL	1 row(s) inserted.	1
110	0	INSERT INTO PRODUCT (PRODUCT_NAME, PRICE, INVENTORY, TAXABL	1 row(s) inserted.	1
111	0	INSERT INTO PRODUCT (PRODUCT_NAME, PRICE, INVENTORY, TAXABL	1 row(s) inserted.	1
112	0.01	INSERT INTO PRODUCT (PRODUCT_NAME, PRICE, INVENTORY, TAXABL	1 row(s) inserted.	1
113	0	INSERT INTO PRODUCT (PRODUCT_NAME, PRICE, INVENTORY, TAXABL	1 row(s) inserted.	1
114	0	INSERT INTO PRODUCT (PRODUCT_NAME, PRICE, INVENTORY, TAXABL	1 row(s) inserted.	1
115	0.01	INSERT INTO PRODUCT (PRODUCT_NAME, PRICE, INVENTORY, TAXABL	1 row(s) inserted.	1
116	0	INSERT INTO PRODUCT (PRODUCT_NAME, PRICE, INVENTORY, TAXABL	1 row(s) inserted.	1
117	0	INSERT INTO PRODUCT (PRODUCT_NAME, PRICE, INVENTORY, TAXABL	1 row(s) inserted.	1
118	0.01	INSERT INTO PRODUCT (PRODUCT_NAME, PRICE, INVENTORY, TAXABL	1 row(s) inserted.	1
119	0	INSERT INTO PRODUCT (PRODUCT_NAME, PRICE, INVENTORY, TAXABL	1 row(s) inserted.	1
120	0	INSERT INTO PRODUCT (PRODUCT_NAME, PRICE, INVENTORY, TAXABL	1 row(s) inserted.	1

121	0.02	INSERT INTO ORDER_HAS_PRODUCTS (PRODUCT_ID, ORDER_ID) VALUES	1 row(s) inserted.	1
122	0	INSERT INTO ORDER_HAS_PRODUCTS (PRODUCT_ID, ORDER_ID) VALUES	1 row(s) inserted.	1
123	0.01	INSERT INTO ORDER_HAS_PRODUCTS (PRODUCT_ID, ORDER_ID) VALUES	1 row(s) inserted.	1
124	0	INSERT INTO ORDER_HAS_PRODUCTS (PRODUCT_ID, ORDER_ID) VALUES	1 row(s) inserted.	1
125	0	INSERT INTO ORDER_HAS_PRODUCTS (PRODUCT_ID, ORDER_ID) VALUES	1 row(s) inserted.	1
126	0	INSERT INTO ORDER_HAS_PRODUCTS (PRODUCT_ID, ORDER_ID) VALUES	1 row(s) inserted.	1
127	0.01	INSERT INTO ORDER_HAS_PRODUCTS (PRODUCT_ID, ORDER_ID) VALUES	1 row(s) inserted.	1
128	0	INSERT INTO ORDER_HAS_PRODUCTS (PRODUCT_ID, ORDER_ID) VALUES	1 row(s) inserted.	1
129	0	INSERT INTO ORDER_HAS_PRODUCTS (PRODUCT_ID, ORDER_ID) VALUES	1 row(s) inserted.	1
130	0	INSERT INTO ORDER_HAS_PRODUCTS (PRODUCT_ID, ORDER_ID) VALUES	1 row(s) inserted.	1
131	0.01	INSERT INTO ORDER_HAS_PRODUCTS (PRODUCT_ID, ORDER_ID) VALUES	1 row(s) inserted.	1
132	0	INSERT INTO ORDER_HAS_PRODUCTS (PRODUCT_ID, ORDER_ID) VALUES	1 row(s) inserted.	1
133	0	INSERT INTO ORDER_HAS_PRODUCTS (PRODUCT_ID, ORDER_ID) VALUES	1 row(s) inserted.	1
134	0	INSERT INTO ORDER_HAS_PRODUCTS (PRODUCT_ID, ORDER_ID) VALUES	1 row(s) inserted.	1

135	0.01	INSERT INTO ORDER_HAS_PRODUCTS (PRODUCT_ID, ORDER_ID) VALUES	1 row(s) inserted.	1
136	0.02	INSERT INTO STORE_SELL_PRODUCTS (STORE_ID, PRODUCT_ID) VALUE	1 row(s) inserted.	1
137	0.01	INSERT INTO STORE_SELL_PRODUCTS (STORE_ID, PRODUCT_ID) VALUE	1 row(s) inserted.	1
138	0	INSERT INTO STORE_SELL_PRODUCTS (STORE_ID, PRODUCT_ID) VALUE	1 row(s) inserted.	1
139	0	INSERT INTO STORE_SELL_PRODUCTS (STORE_ID, PRODUCT_ID) VALUE	1 row(s) inserted.	1
140	0.01	INSERT INTO STORE_SELL_PRODUCTS (STORE_ID, PRODUCT_ID) VALUE	1 row(s) inserted.	1
141	0	INSERT INTO STORE_SELL_PRODUCTS (STORE_ID, PRODUCT_ID) VALUE	1 row(s) inserted.	1
142	0	INSERT INTO STORE_SELL_PRODUCTS (STORE_ID, PRODUCT_ID) VALUE	1 row(s) inserted.	1
143	0.01	INSERT INTO STORE_SELL_PRODUCTS (STORE_ID, PRODUCT_ID) VALUE	1 row(s) inserted.	1
144	0	INSERT INTO STORE_SELL_PRODUCTS (STORE_ID, PRODUCT_ID) VALUE	1 row(s) inserted.	1
145	0	INSERT INTO STORE_SELL_PRODUCTS (STORE_ID, PRODUCT_ID) VALUE	1 row(s) inserted.	1
146	0.01	INSERT INTO STORE_SELL_PRODUCTS (STORE_ID, PRODUCT_ID) VALUE	1 row(s) inserted.	1
147	0	INSERT INTO STORE_SELL_PRODUCTS (STORE_ID, PRODUCT_ID) VALUE	1 row(s) inserted.	1
148	0	INSERT INTO STORE_SELL_PRODUCTS (STORE_ID, PRODUCT_ID) VALUE	1 row(s) inserted.	1

149	0.01	INSERT INTO STORE_SELL_PRODUCTS (STORE_ID, PRODUCT_ID) VALUE	1 row(s) inserted.	1
150	0	INSERT INTO STORE_SELL_PRODUCTS (STORE_ID, PRODUCT_ID) VALUE	1 row(s) inserted.	1

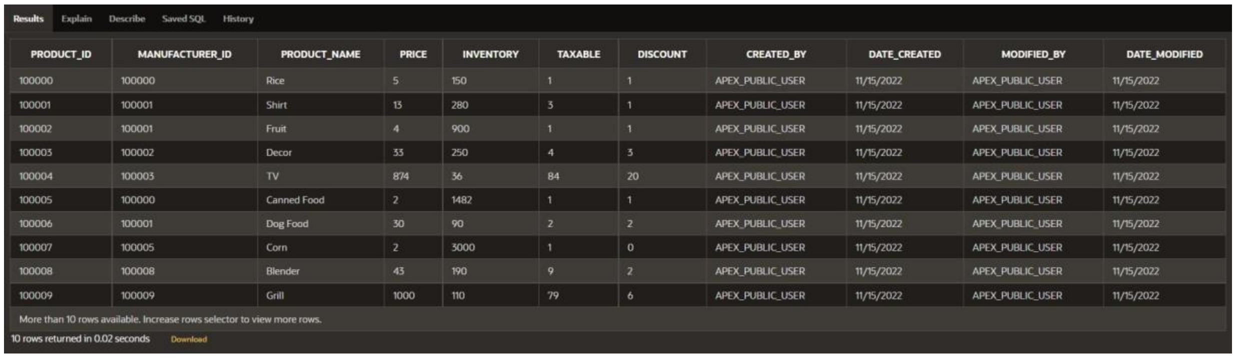
SQL QUERY OUTPUT:



FIRST_NAME	LAST_NAME	DOB	EMAIL	PHONE
Rachel	Pham	09/05/1965	rachelpham@gmail.com	4694283574
Mihir	Harvi	10/25/1975	mihir@gmail.com	2313213123
Kayla	Smith	04/15/1995	kayla@gmail.com	4445556986
Riley	Frost	11/28/1997	riley@gmail.com	1238870962
Jodie	Pham	08/05/2005	jodie@gmail.com	4242447898
Love	Smith	10/06/1955	lovesmith@gmail.com	2223334675
Bruce	Lee	12/18/2001	brucelee@gmail.com	2342227878
Gasan	Elkhodari	09/05/1995	gelkhodari@gmail.com	2314343231
Bentley	John	08/15/1969	bentley@gmail.com	1212334213
Dave	Treck	12/11/1965	dave@gmail.com	2323124421

More than 10 rows available. Increase rows selector to view more rows.

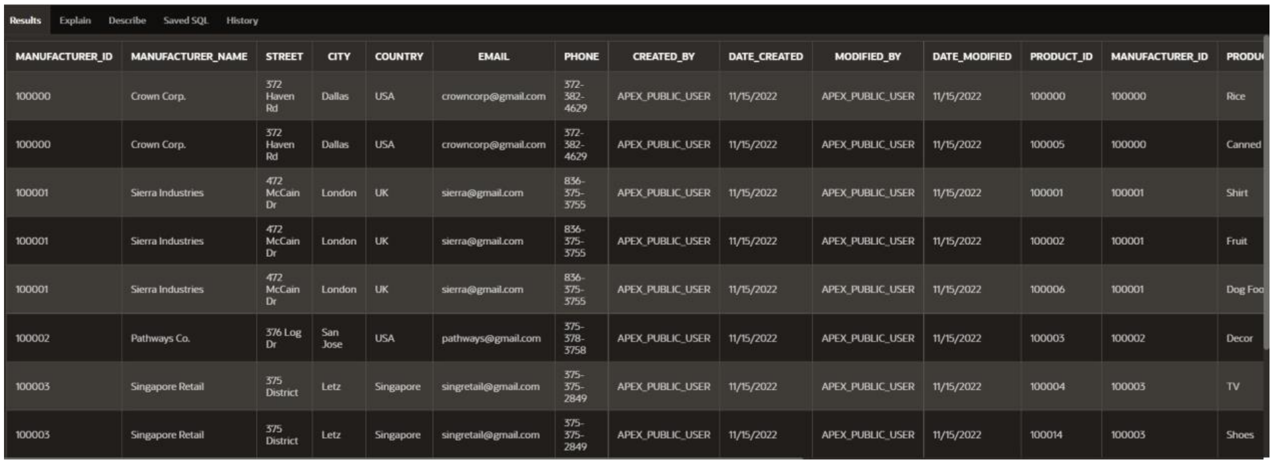
10 rows returned in 0.01 seconds



PRODUCT_ID	MANUFACTURER_ID	PRODUCT_NAME	PRICE	INVENTORY	TAXABLE	DISCOUNT	CREATED_BY	DATE_CREATED	MODIFIED_BY	DATE_MODIFIED
100000	100000	Rice	5	150	1	1	APEX_PUBLIC_USER	11/15/2022	APEX_PUBLIC_USER	11/15/2022
100001	100001	Shirt	13	280	3	1	APEX_PUBLIC_USER	11/15/2022	APEX_PUBLIC_USER	11/15/2022
100002	100001	Fruit	4	900	1	1	APEX_PUBLIC_USER	11/15/2022	APEX_PUBLIC_USER	11/15/2022
100003	100002	Decor	33	250	4	3	APEX_PUBLIC_USER	11/15/2022	APEX_PUBLIC_USER	11/15/2022
100004	100003	TV	874	36	84	20	APEX_PUBLIC_USER	11/15/2022	APEX_PUBLIC_USER	11/15/2022
100005	100000	Canned Food	2	1482	1	1	APEX_PUBLIC_USER	11/15/2022	APEX_PUBLIC_USER	11/15/2022
100006	100001	Dog Food	30	90	2	2	APEX_PUBLIC_USER	11/15/2022	APEX_PUBLIC_USER	11/15/2022
100007	100005	Corn	2	3000	1	0	APEX_PUBLIC_USER	11/15/2022	APEX_PUBLIC_USER	11/15/2022
100008	100008	Blender	43	190	9	2	APEX_PUBLIC_USER	11/15/2022	APEX_PUBLIC_USER	11/15/2022
100009	100009	Grill	1000	110	79	6	APEX_PUBLIC_USER	11/15/2022	APEX_PUBLIC_USER	11/15/2022

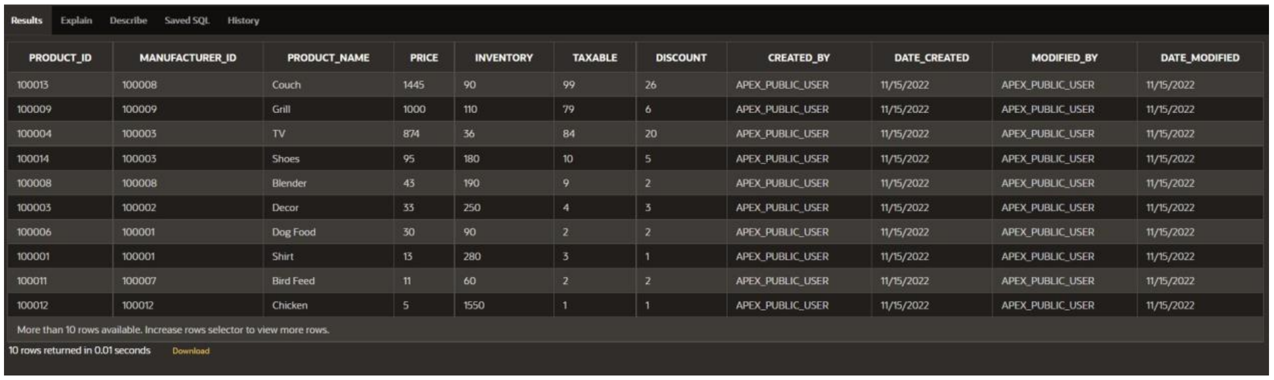
More than 10 rows available. Increase rows selector to view more rows.

10 rows returned in 0.02 seconds



Results Explain Describe Saved SQL History

MANUFACTURER_ID	MANUFACTURER_NAME	STREET	CITY	COUNTRY	EMAIL	PHONE	CREATED_BY	DATE_CREATED	MODIFIED_BY	DATE_MODIFIED	PRODUCT_ID	MANUFACTURER_ID	PRODU
100000	Crown Corp.	372 Haven Rd	Dallas	USA	crowncorp@gmail.com	372-382-4629	APEX_PUBLIC_USER	11/15/2022	APEX_PUBLIC_USER	11/15/2022	100000	100000	Rice
100000	Crown Corp.	372 Haven Rd	Dallas	USA	crowncorp@gmail.com	372-382-4629	APEX_PUBLIC_USER	11/15/2022	APEX_PUBLIC_USER	11/15/2022	100005	100000	Canned
100001	Sierra Industries	472 McCain Dr	London	UK	sierra@gmail.com	836-375-3755	APEX_PUBLIC_USER	11/15/2022	APEX_PUBLIC_USER	11/15/2022	100001	100001	Shirt
100001	Sierra Industries	472 McCain Dr	London	UK	sierra@gmail.com	836-375-3755	APEX_PUBLIC_USER	11/15/2022	APEX_PUBLIC_USER	11/15/2022	100002	100001	Fruit
100001	Sierra Industries	472 McCain Dr	London	UK	sierra@gmail.com	836-375-3755	APEX_PUBLIC_USER	11/15/2022	APEX_PUBLIC_USER	11/15/2022	100006	100001	Dog Foo
100002	Pathways Co.	376 Log Dr	San Jose	USA	pathways@gmail.com	375-378-3758	APEX_PUBLIC_USER	11/15/2022	APEX_PUBLIC_USER	11/15/2022	100003	100002	Decor
100003	Singapore Retail	375 District	Letz	Singapore	singretail@gmail.com	375-375-2849	APEX_PUBLIC_USER	11/15/2022	APEX_PUBLIC_USER	11/15/2022	100004	100003	TV
100003	Singapore Retail	375 District	Letz	Singapore	singretail@gmail.com	375-375-2849	APEX_PUBLIC_USER	11/15/2022	APEX_PUBLIC_USER	11/15/2022	100014	100003	Shoes

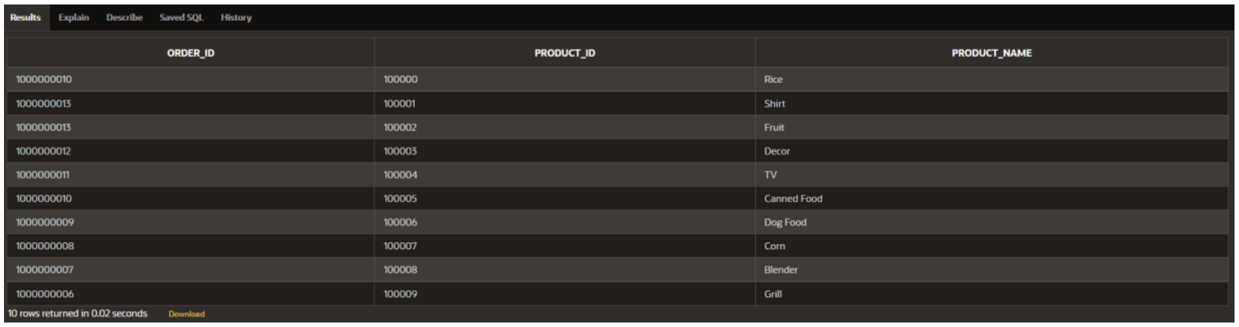


Results Explain Describe Saved SQL History

PRODUCT_ID	MANUFACTURER_ID	PRODUCT_NAME	PRICE	INVENTORY	TAXABLE	DISCOUNT	CREATED_BY	DATE_CREATED	MODIFIED_BY	DATE_MODIFIED
100013	100008	Couch	1445	90	99	26	APEX_PUBLIC_USER	11/15/2022	APEX_PUBLIC_USER	11/15/2022
100009	100009	Grill	1000	110	79	6	APEX_PUBLIC_USER	11/15/2022	APEX_PUBLIC_USER	11/15/2022
100004	100003	TV	874	36	84	20	APEX_PUBLIC_USER	11/15/2022	APEX_PUBLIC_USER	11/15/2022
100014	100003	Shoes	95	180	10	5	APEX_PUBLIC_USER	11/15/2022	APEX_PUBLIC_USER	11/15/2022
100008	100008	Blender	43	190	9	2	APEX_PUBLIC_USER	11/15/2022	APEX_PUBLIC_USER	11/15/2022
100003	100002	Decor	33	250	4	3	APEX_PUBLIC_USER	11/15/2022	APEX_PUBLIC_USER	11/15/2022
100006	100001	Dog Food	30	90	2	2	APEX_PUBLIC_USER	11/15/2022	APEX_PUBLIC_USER	11/15/2022
100001	100001	Shirt	13	280	3	1	APEX_PUBLIC_USER	11/15/2022	APEX_PUBLIC_USER	11/15/2022
100011	100007	Bird Feed	11	60	2	2	APEX_PUBLIC_USER	11/15/2022	APEX_PUBLIC_USER	11/15/2022
100012	100012	Chicken	5	1550	1	1	APEX_PUBLIC_USER	11/15/2022	APEX_PUBLIC_USER	11/15/2022

More than 10 rows available. Increase rows selector to view more rows.

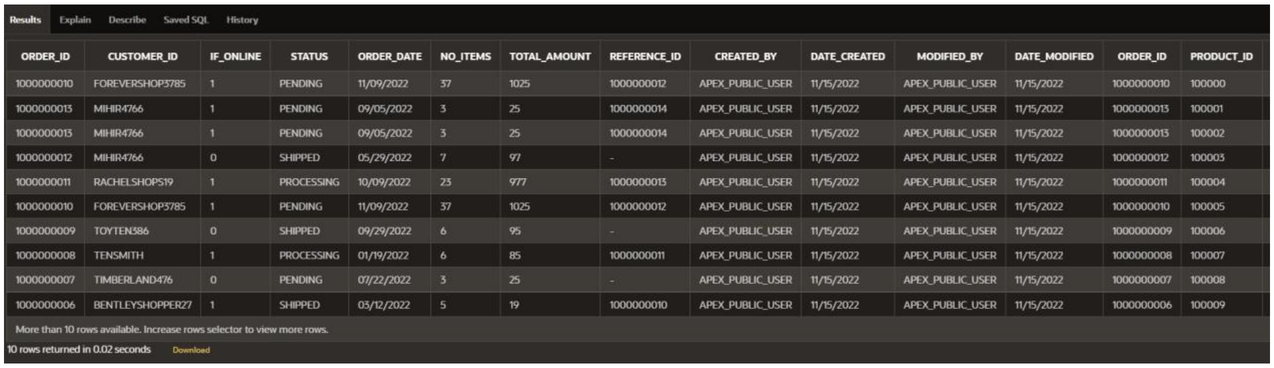
10 rows returned in 0.01 seconds Download



Results Explain Describe Saved SQL History

ORDER_ID	PRODUCT_ID	PRODUCT_NAME
1000000010	100000	Rice
1000000013	100001	Shirt
1000000013	100002	Fruit
1000000012	100003	Decor
1000000011	100004	TV
1000000010	100005	Canned Food
1000000009	100006	Dog Food
1000000008	100007	Corn
1000000007	100008	Blender
1000000006	100009	Grill

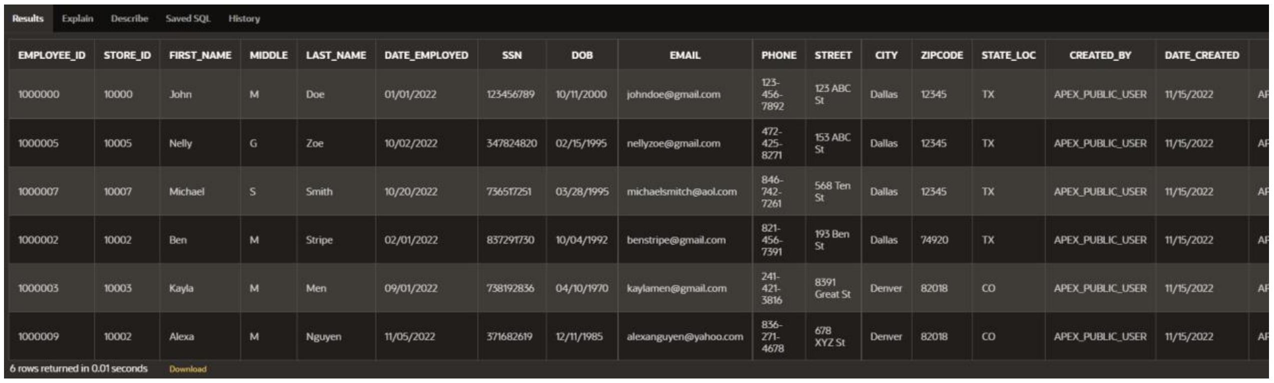
10 rows returned in 0.02 seconds Download



ORDER_ID	CUSTOMER_ID	IF_ONLINE	STATUS	ORDER_DATE	NO_ITEMS	TOTAL_AMOUNT	REFERENCE_ID	CREATED_BY	DATE_CREATED	MODIFIED_BY	DATE_MODIFIED	ORDER_ID	PRODUCT_ID
1000000010	FOREVERSHOP3785	1	PENDING	11/09/2022	37	1025	1000000012	APEX_PUBLIC_USER	11/15/2022	APEX_PUBLIC_USER	11/15/2022	1000000010	100000
1000000013	MIHIR4766	1	PENDING	09/05/2022	3	25	1000000014	APEX_PUBLIC_USER	11/15/2022	APEX_PUBLIC_USER	11/15/2022	1000000013	100001
1000000013	MIHIR4766	1	PENDING	09/05/2022	3	25	1000000014	APEX_PUBLIC_USER	11/15/2022	APEX_PUBLIC_USER	11/15/2022	1000000013	100002
1000000012	MIHIR4766	0	SHIPPED	05/29/2022	7	97	-	APEX_PUBLIC_USER	11/15/2022	APEX_PUBLIC_USER	11/15/2022	1000000012	100003
1000000011	RACHELSHOPS19	1	PROCESSING	10/09/2022	23	977	1000000013	APEX_PUBLIC_USER	11/15/2022	APEX_PUBLIC_USER	11/15/2022	1000000011	100004
1000000010	FOREVERSHOP3785	1	PENDING	11/09/2022	37	1025	1000000012	APEX_PUBLIC_USER	11/15/2022	APEX_PUBLIC_USER	11/15/2022	1000000010	100005
1000000009	TOYTEN386	0	SHIPPED	09/29/2022	6	95	-	APEX_PUBLIC_USER	11/15/2022	APEX_PUBLIC_USER	11/15/2022	1000000009	100006
1000000008	TENSMITH	1	PROCESSING	01/19/2022	6	85	1000000011	APEX_PUBLIC_USER	11/15/2022	APEX_PUBLIC_USER	11/15/2022	1000000008	100007
1000000007	TIMBERLAND476	0	PENDING	07/22/2022	3	25	-	APEX_PUBLIC_USER	11/15/2022	APEX_PUBLIC_USER	11/15/2022	1000000007	100008
1000000006	BENTLEYSHOPPER27	1	SHIPPED	03/12/2022	5	19	1000000010	APEX_PUBLIC_USER	11/15/2022	APEX_PUBLIC_USER	11/15/2022	1000000006	100009

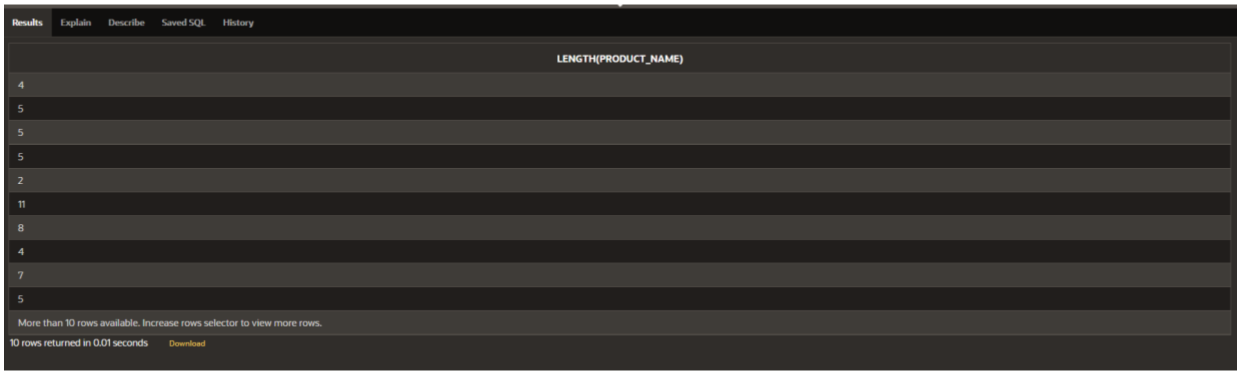
More than 10 rows available. Increase rows selector to view more rows.

10 rows returned in 0.02 seconds Download



EMPLOYEE_ID	STORE_ID	FIRST_NAME	MIDDLE	LAST_NAME	DATE_EMPLOYED	SSN	DOB	EMAIL	PHONE	STREET	CITY	ZIPCODE	STATE_LOC	CREATED_BY	DATE_CREATED
1000000	10000	John	M	Doe	01/01/2022	123456789	10/11/2000	johndoe@gmail.com	123-456-7892	123 ABC St	Dallas	12345	TX	APEX_PUBLIC_USER	11/15/2022
1000005	10005	Nelly	G	Zoe	10/02/2022	347824820	02/15/1995	nellyzoe@gmail.com	472-425-8271	153 ABC St	Dallas	12345	TX	APEX_PUBLIC_USER	11/15/2022
1000007	10007	Michael	S	Smith	10/20/2022	736517251	03/28/1995	michaelsmitch@aol.com	846-742-7261	568 Ten St	Dallas	12345	TX	APEX_PUBLIC_USER	11/15/2022
1000002	10002	Ben	M	Stripe	02/01/2022	837291730	10/04/1992	benstripe@gmail.com	821-456-7391	193 Ben St	Dallas	74920	TX	APEX_PUBLIC_USER	11/15/2022
1000003	10003	Kayla	M	Men	09/01/2022	738192836	04/10/1970	kaylamen@gmail.com	241-421-3816	8391 Great St	Denver	82018	CO	APEX_PUBLIC_USER	11/15/2022
1000009	10002	Alexa	M	Nguyen	11/05/2022	371682619	12/11/1985	alexanguyen@yahoo.com	836-271-4678	678 XYZ St	Denver	82018	CO	APEX_PUBLIC_USER	11/15/2022

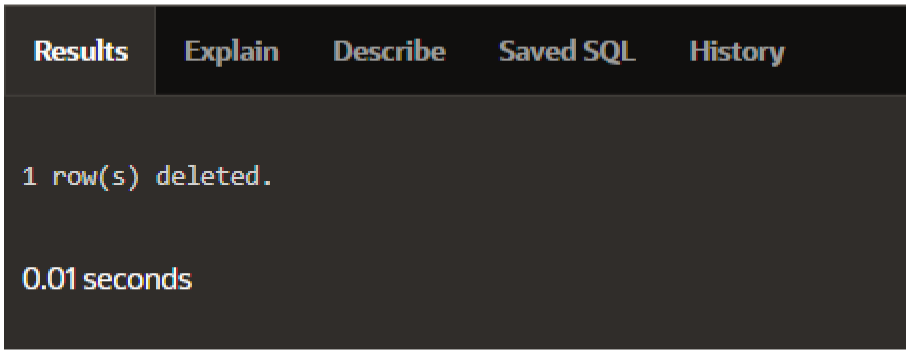
6 rows returned in 0.01 seconds Download



LENGTH(PRODUCT_NAME)
4
5
5
5
2
11
8
4
7
5

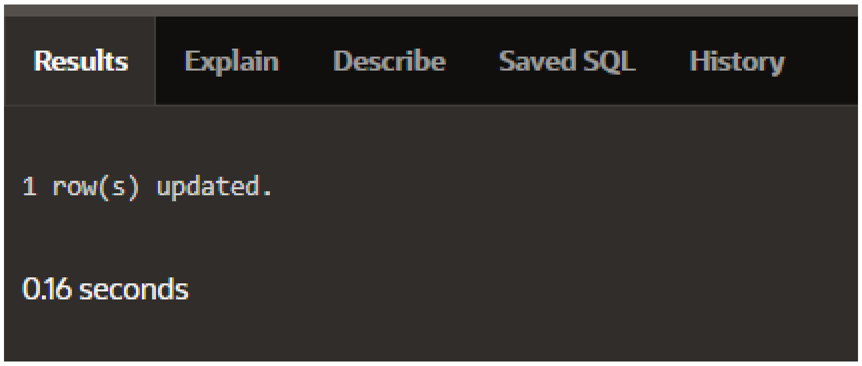
More than 10 rows available. Increase rows selector to view more rows.

10 rows returned in 0.01 seconds Download



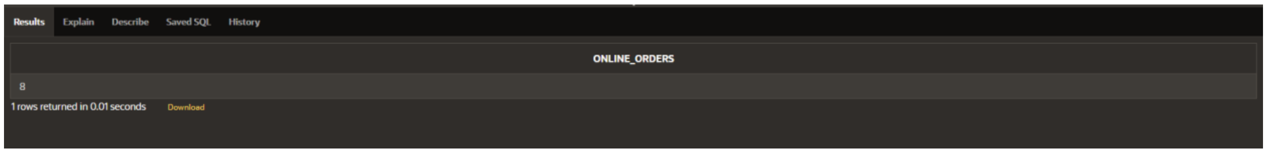
```
1 row(s) deleted.

0.01 seconds
```



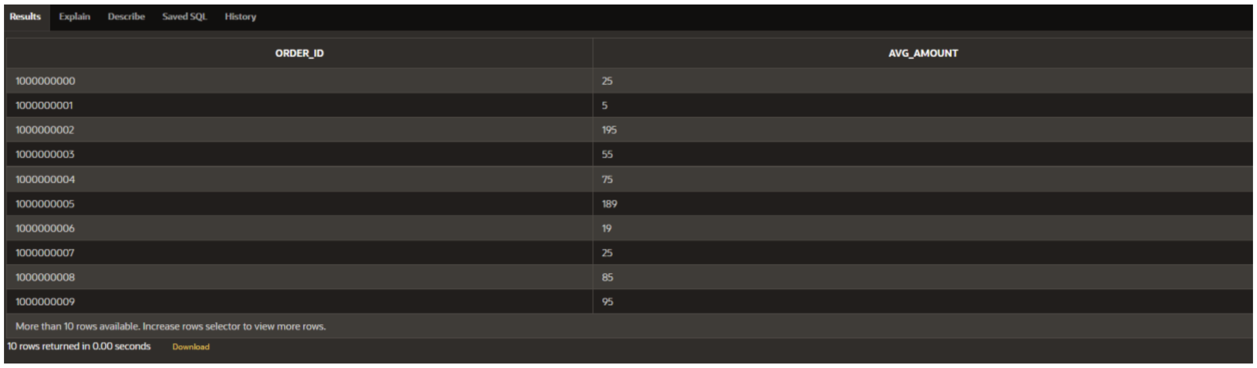
```
1 row(s) updated.

0.16 seconds
```



ONLINE_ORDERS
8

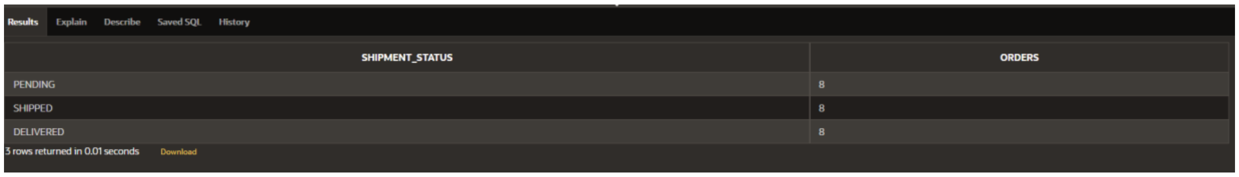
1 rows returned in 0.01 seconds



ORDER_ID	AVG_AMOUNT
1000000000	25
1000000001	5
1000000002	195
1000000003	55
1000000004	75
1000000005	189
1000000006	19
1000000007	25
1000000008	85
1000000009	95

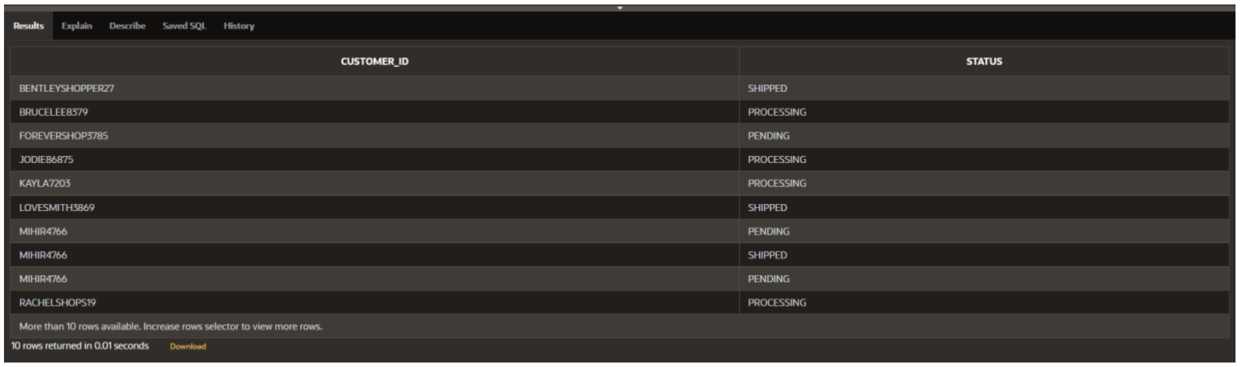
More than 10 rows available. Increase rows selector to view more rows.

10 rows returned in 0.00 seconds



SHIPMENT_STATUS	ORDERS
PENDING	8
SHIPPED	8
DELIVERED	8

3 rows returned in 0.01 seconds

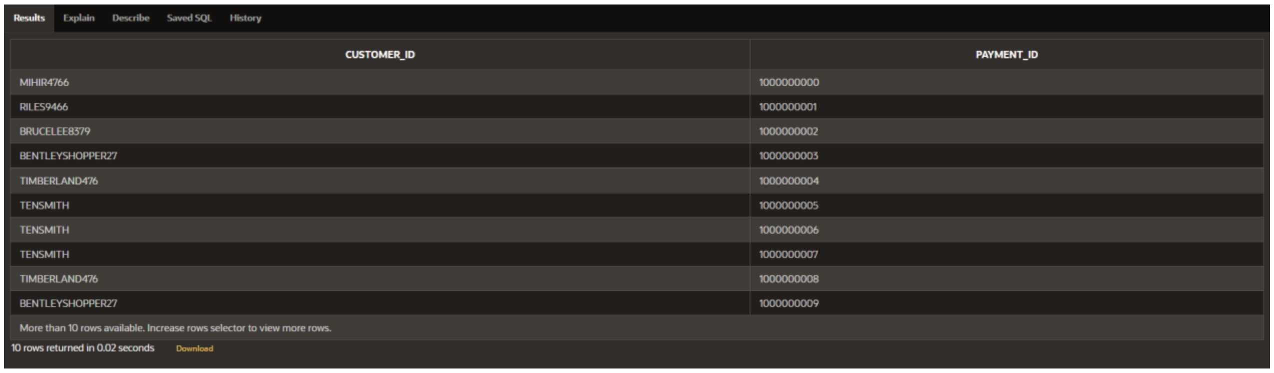


Results Explain Describe Saved SQL History

CUSTOMER_ID	STATUS
BENTLEYSHOPPER27	SHIPPED
BRUCELEE8379	PROCESSING
FOREVERSHOP3785	PENDING
JODIE86875	PROCESSING
KAYLA7203	PROCESSING
LOVESMITH3869	SHIPPED
MIHIR4766	PENDING
MIHIR4766	SHIPPED
MIHIR4766	PENDING
RACHELSHOPS19	PROCESSING

More than 10 rows available. Increase rows selector to view more rows.

10 rows returned in 0.01 seconds Download

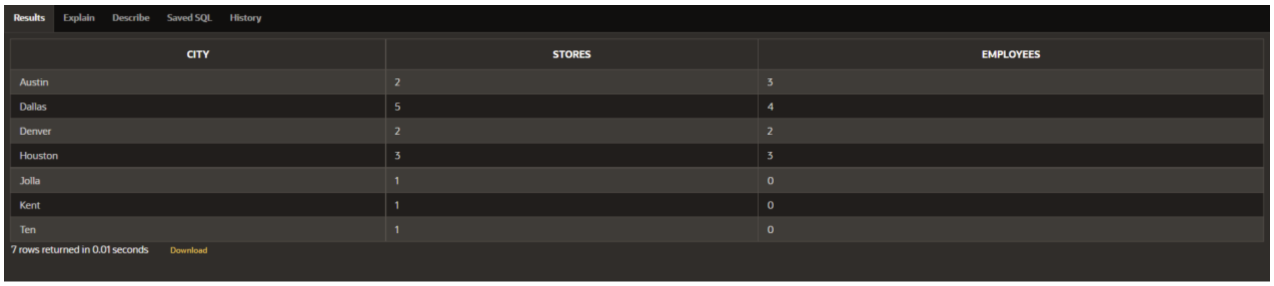


Results Explain Describe Saved SQL History

CUSTOMER_ID	PAYMENT_ID
MIHIR4766	1000000000
RILES9466	1000000001
BRUCELEE8379	1000000002
BENTLEYSHOPPER27	1000000003
TIMBERLAND476	1000000004
TENSMITH	1000000005
TENSMITH	1000000006
TENSMITH	1000000007
TIMBERLAND476	1000000008
BENTLEYSHOPPER27	1000000009

More than 10 rows available. Increase rows selector to view more rows.

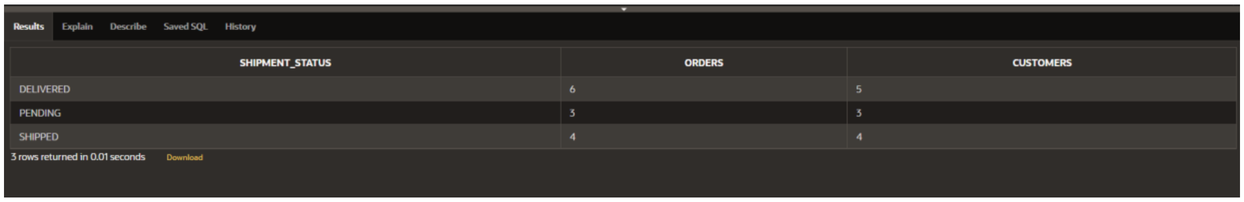
10 rows returned in 0.02 seconds Download



Results Explain Describe Saved SQL History

CITY	STORES	EMPLOYEES
Austin	2	3
Dallas	5	4
Denver	2	2
Houston	3	3
Jolla	1	0
Kent	1	0
Ten	1	0

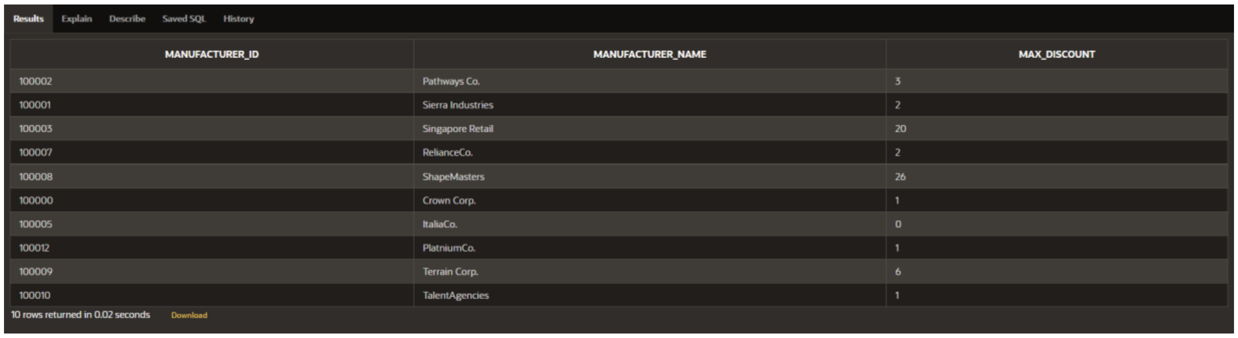
7 rows returned in 0.01 seconds Download



Results Explain Describe Saved SQL History

SHIPMENT_STATUS	ORDERS	CUSTOMERS
DELIVERED	6	5
PENDING	3	3
SHIPPED	4	4

3 rows returned in 0.01 seconds Download



MANUFACTURER_ID	MANUFACTURER_NAME	MAX_DISCOUNT
100002	Pathways Co.	3
100001	Sierra Industries	2
100003	Singapore Retail	20
100007	RelianceCo.	2
100008	ShapeMasters	26
100000	Crown Corp.	1
100005	ItaliaCo.	0
100012	PlatniumCo.	1
100009	Terrain Corp.	6
100010	TalentAgencies	1

10 rows returned in 0.02 seconds

DATABASE ADMINISTRATION AND MONITORING

1.1 Roles and Responsibilities

- Database Administrator: The Database Administrator, and supporting database staff, will direct upkeep of the database and the advancement of new SQL contents to help evolving necessities.
- System Administrator: The Database Administrator and supporting staff will keep up with the condition of the server running the DBMS, including the DBMS software itself, the server working framework, and any supporting devices.
- Security Administrator: The security administrator and other safety crew will keep up with the trustworthiness of the safety efforts and frameworks encompassing the database and will work straightforwardly with the other organization groups to direct the overhaul of server programming and the adjustment of the database and SQL scripts in reactions to security issues and changes in security strategy.

1.2 System Information

DBMS: Oracle Apex

System requirements: Internet Connectivity, Internet browser.

1.3 Performance Monitoring and Database Efficiency

Execution observing and support of the information base, the DBMS, and the servers running it will be a joint liability between the database administration and system administration teams. The database administration group will be answerable for checking and keeping up with the information base, while the system administration group will be liable for the servers and supporting programming. Keeping up with the actual DBMS will be taken care of by the two groups.

1.4 Data Formats

The database, as by and by arranged, requires data move of three sorts: string, integer, time, and date information as binary data; image transfer as Portable Network Graphics (PNG) files; and level data move as a proprietary level (.lvl) format used exclusively close to the end client. The raw binary data will be stored in the database directly and transferred by the DBMS; while the limit and move of picture and level data will be regulated by an alternate file storing system, which the database will associate with through URIs featuring unequivocal records.

1.5 Backup and Recovery

Because of the normal recurrence of changes made to the database as new and refreshed level information, client enlistments, and relics from end client connection, delta reinforcements of the database will be performed two times everyday, and a forming framework will briefly store a record of changes as they are made. Full reinforcements of the database will be performed during a week by week upkeep period at 3 AM EST each Tuesday.

REFERENCES

Coronel, C., & Morris, S. A. (2019). *Database Systems Design, implementation, and management*. Cengage.

Martin, S. (2022, August 23). *What's The Best Data Source? retailer direct or syndicated Nielsen/IRI data?* CPG Data Tip Sheet. Retrieved November 14, 2022, from https://www.cpgdatainsights.com/get-started-with-nielsen-iri/data-source-retailer-syndicated/