# GROCERY MANAGEMENT SYSTEM

**DBMS PROJECT REPORT**

**Submitted to:Sumit Sharma**

**GROUP-2CO6**

THAPAR INSTITUTE
OF ENGINEERING & TECHNOLOGY
(Deemed to be University)


| | |
|---|---|
| **102103174** | **JAPJOT SINGH** |
| **102103167** | **RAJVIR SINGH** |
| **102103175** | **SANDALPREET KAUR** |

# TABLE OF CONTENTS

# INTRODUCTION

Grocery Management System is an application designed to manage the operations of a grocery store using SQL. This system provides an efficient and organized way of managing inventory, sales, and customer data. It enables store owners to keep track of their inventory levels, manage their sales, and provide an excellent customer experience. The grocery store management system requires a database that can store customer information, employee details, store information, inventory details, and checkout information. The system must provide functionalities for adding new customers, employees, stores, and items to the inventory.

## NEED OF PROJECT

The grocery management system is needed to streamline the operations of a grocery store and improve store performance. With real-time data on inventory levels, sales, and customer data, managers can make informed decisions about stock levels, pricing, and promotions. The system also helps to improve customer satisfaction by providing a fast and efficient checkout process and personalized shopping experience.

## SCOPE OF PROJECT

This project aims to develop a MySQL database management system for a grocery store. The database will consist of several tables, including customers, employees, stores, inventory, and checkout. By using this system, the grocery store will be able to efficiently manage its inventory, track sales, and customer information, and provide better customer service. The system should be scalable to handle the increasing demands of a growing store, and it should be easy to integrate with other systems, such as accounting software and point-of-sale systems. Finally, the system should be easy to maintain and update to keep up with changing business needs.

## REQUIREMENT ANALYSIS

The Grocery Management System is developed to help store owners in managing their grocery stores efficiently. This system has various features that are essential for effective store management, such as inventory management, employee management, customer management, and checkout management. It should also enable employees to manage inventory levels, update item details, and track sales. Customers should be able to view their purchase history, and the store should be able to analyze sales data to make informed decisions about inventory management.

Employee Management

The system also allows the store owner to manage the employees efficiently. The employees table contains details of all the employees, including their names, contact information, social security numbers, and employment dates. The system enables the owner to keep track of employee attendance, salary payments, and performance evaluations.

Customer Management

The customer management feature of the system allows the store owner to manage the customers' information effectively. The customers table contains customer details such as their names, phone numbers, and email addresses. The system enables the owner to keep track of customers' purchase history, making it easier to personalize their shopping experience.

Checkout Management

The system provides an efficient and effective means of managing the checkout process. The checkout table contains details of all the transactions made, including the customer details, store details, employee details, and transaction details. The system enables the owner to keep track of sales, revenue, and profits.

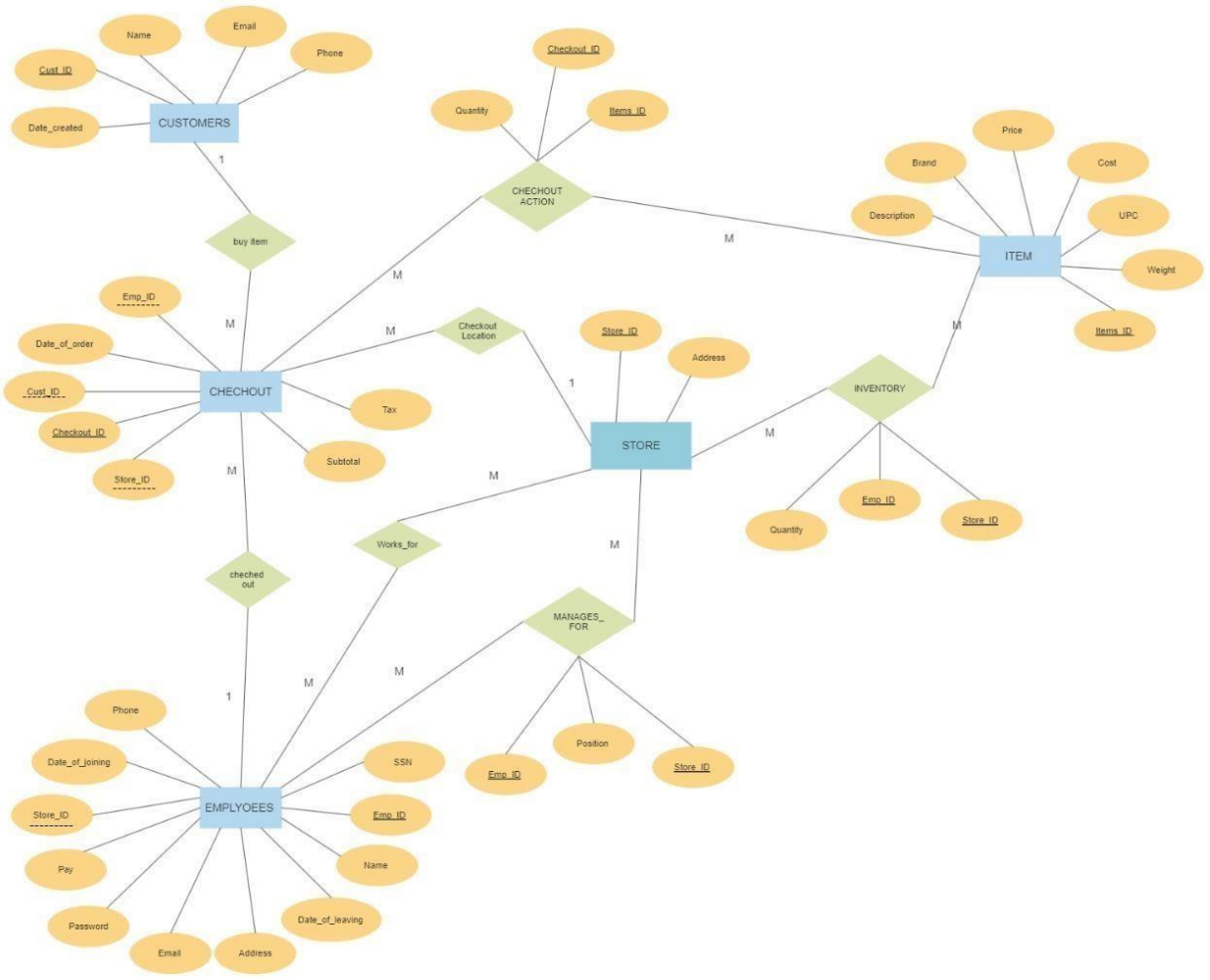
Item Management

The system should allow grocery stores to manage item data, such as itemdescription, brand, cost, price, weight, and UPC. The system should also allow items to be uniquely identified by their UPC.

Inventory Management

The system should allow grocery stores to manage inventory data, such as items ID, store ID, and quantity. The system should also allow the tracking of items instock at each store. The inventory management feature of the system allows the store owner tomanage the inventory of the store effectively. The inventory table contains the items in stock, their quantities, and their respective stores. The system enables the owner to track the stock levels, making it easier to restock the shelves when required.

# ER-DIAGRAM


Name
Email
Phone
Cust_ID
Date_created
CUSTOMERS
1
buy item
Checkout_ID
Quantity
Items_ID
CHECHOUT ACTION
M
Price
Brand
Cost
Description
UPC
ITEM
Weight
Items_ID
Emp_ID
Date_of_order
Cust_ID
Checkout_ID
Store_ID
CHECHOUT
Tax
Subtotal
Checkout Location
Store_ID
Address
STORE
INVENTORY
Quantity
Emp_ID
Store_ID
Works_for
cheched out
MANAGES_ FOR
Position
Emp_ID
Store_ID
Phone
Date_of_joining
Store_ID
Pay
Password
Email
Address
Date_of_leaving
Name
Emp_ID
SSN
EMPLYOEES

# ER TO TABLES

## CUSTOMERS

| Cust_ID | Name | Phone | Email | Date_created |
|---|---|---|---|---|

1. Cust_ID : integer Id (Primary key)
2. Name: 128-letter string character ranged ‘a’ to ‘z’
3. Phone: 10-digit string character ranged ‘0’ to ‘9’
4. Email: 128-letter string character ranged ‘a’ to ‘z’ plus ‘@’ symbol
5. Date_created: valid date

## CHECKOUT

| Checkout_ID | Cust_ID | Store_ID | Emp_ID | Date_of_order | Subtotal | Tax |
|---|---|---|---|---|---|---|

1. Checkout_ID: auto-generate integer ID (primary key)
2. Cust_ID: unique integer number identifying each store – Foreign Key from CUSTOMERS entity
3. Store_ID: unique integer number identifying each store – Foreign Key from STORE entity
4. Emp_ID: unique integer number identifying each store – Foreign Key from EMPLOYEE entity
5. Date_of_order: valid date type
6. Subtotal: double with two digits precision
7. Tax: double with two digits precision

## CHECKOUTACTION

| Checkout_ID | Items_ID | Quantity |
|---|---|---|

1. Checkout_ID: auto-generated integer ID- Foreign Key from CHECKOUT entity
2. Items_ID; auto-generated integer ID- Foreign Key from ITEMS entity
3. Quantity: integer describing number of items purchased on that transaction.

## ITEMS

| Items_ID | Description | Brand | Cost | Price | Weight | UPC |
|---|---|---|---|---|---|---|

1. Items_ID: auto-generated integer ID (Primary key)
2. Description: 128-letter character string ranged 'a' through each plus special characters
3. Brand: 32-letter character string ranged a through 2 each plus special characters
4. Cost: factory cost of item
5. Price: valid positive double with two digit precision
6. Weight: valid positive double with two digit precision
7. UPC: valid positive integer

## EMPLOYEES

| Emp_ID | Name | SSN | Phone | Address | Pay | Store_ID | Date_of_joining | Date_of_leaving | Email | Password |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | |

1. Emp ID: auto-generated integer ID
2. Name: 128-letter character string ranged "a' through “z” each
3. SSN: 12 digit character integer ranged "0" through "9" (Social Security number)
4. Phone: 10-digt character integer ranged "0" through "9"
5. Address: physical address of employee
6. Password: encrypted password field
7. Store ID: unique integer number identifying each store - Foreign Key from STORE entity
8. Date_of_joining: valid Date hired
9. Date_of_leaving: valid Date employment done. If NULL still employed.
10. Pay- integer amount employee makes either annual or hour based on pay type
11. Email- 128-letter character string ranged 'a' through

## STORE

| Store_ID | Address |
|---|---|

1. Store_ID: auto-generated integer ID (Primary key)
2. Address: 128-letter string character ranged ‘a’ to ‘z’ plus a special character

## INVENTORY

| Store_ID | Items_ID | Quantity |
|---|---|---|

1. Items_ID: auto-generated integer ID- Foreign Key from ITEMS entity
2. Store_ID: auto-generated integer ID- Foreign Key from STORE entity Quantity valid positive integer
3. Quantity: valid positive integer

## MANAGES_FOR

| Emp_ID | Store_ID | Position |
|---|---|---|

1. Emp_ID: auto-generated integer ID – Foreign Key from EMPLOYEES entity
2. Store_ID: auto-generated integer ID- Foreign Key from STORE entity
3. Position: string position of the manager

# NORMALIZATION

**1NF (First Normal Form)**
First Normal Form (1NF) is the first step in the normalization process of a relational database. It requires that the values in each column of a table are atomic, meaning they cannot be further divided into smaller pieces or attributes. In other words, each cell of a table must hold only one value, and that value must be of the same data type I.e no multivalued function exists.

CUSTOMERS

| Cust_ID | Name | Phone | Email | Date_created |
|---|---|---|---|---|

CHECKOUT

| Checkout_ID | Cust_ID | Store_ID | Emp_ID | Date_of_order | Subtotal | Tax |
|---|---|---|---|---|---|---|

CHECKOUTACTION

| Checkout_ID | Items_ID | Quantity |
|---|---|---|

ITEMS

| Items_ID | Description | Brand | Cost | Price | Weight | UPC |
|---|---|---|---|---|---|---|

EMPLOYEES

| Emp_ID | Name | SSN | Phone | Address | Pay | Store_ID | Date_of_joining | Date_of_leaving | Email | Password |
|---|---|---|---|---|---|---|---|---|---|---|

STORE

| Store_ID | Address |
|---|---|

INVENTORY

| Store_ID | Items_ID | Quantity |
|---|---|---|

MANAGES_FOR

| Emp_ID | Store_ID | Position |
|---|---|---|

## 2NF (Second Normal Form)
A table is considered to be in 2NF if it meets the following conditions.

- It is in 1NF.
- It contains no partial dependencies. In other words, 2NF requires that each non-key attribute in a table is dependent on the entire primary key, and not just on part of it. If there are any partial dependencies, they must be removed by splitting the table into multiple tables, each with a separate primary key.

CUSTOMERS

| Cust_ID | Name | Phone | Email | Date_created |
|---|---|---|---|---|

CHECKOUT

| | | | | | | |
|---|---|---|---|---|---|---|
| Checkout_ID | Cust_ID | Store_ID | Emp_ID | Date_of_order | Subtotal | Tax |

CHECKOUTACTION

| | | |
|---|---|---|
| Checkout_ID | Items_ID | Quantity |

ITEMS

| | | | | | | |
|---|---|---|---|---|---|---|
| Items_ID | Description | Brand | Cost | Price | Weight | UPC |

EMPLOYEES

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Emp_ID | Name | SSN | Phone | Address | Pay | Store_ID | Date_of_joining | Date_of_leaving | Email | Password |

STORE

| | |
|---|---|
| Store_ID | Address |

INVENTORY

| | | |
|---|---|---|
| Store_ID | Items_ID | Quantity |

MANAGES_FOR

| | | |
|---|---|---|
| Emp_ID | Store_ID | Position |

## 3NF (Third Normal Form)

A table is in 3NF if it satisfies the following conditions:

- It is in 2NF.
- It does not contain transitive functional dependencies, i.e., a non-prime attribute should not depend on another non-prime attribute in the same table. This reduces data redundancy and improves data integrity.

CUSTOMERS

| | | | | |
|---|---|---|---|---|
| Cust_ID | Name | Phone | Email | Date_created |

CHECKOUT

| | | | | | | |
|---|---|---|---|---|---|---|
| Checkout_ID | Cust_ID | Store_ID | Emp_ID | Date_of_order | Subtotal | Tax |

CHECKOUTACTION

| | | |
|---|---|---|
| Checkout_ID | Items_ID | Quantity |

ITEMS

| | | | | | | |
|---|---|---|---|---|---|---|
| Items_ID | Description | Brand | Cost | Price | Weight | UPC |

EMPLOYEES

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Emp_ID | Name | SSN | Phone | Address | Pay | Store_ID | Date_of_joining | Date_of_leaving | Email | Password |

STORE

| Store_ID | Address |
|---|---|

INVENTORY

| Store_ID | Items_ID | Quantity |
|---|---|---|

MANAGES_FOR

| Emp_ID | Store_ID | Position |
|---|---|---|
| | | |

### 4NF (Fourth Normal Form)

A table is said to be in 4NF if it has no multi-valued dependencies. It is an extension of the third normal form, which deals with the elimination of repeating groups and the prevention of insertion, update, and deletion anomalies. The 4NF is a higher level of normalization that aims to remove redundancy from the database design. Hence, the tables are in 4NF.

### BCNF (Boyce-Codd Normal Form)

A table is in BCNF if and only if every determinant in the table is a candidate key. BCNF is a higher level of normalization that guarantees that every non-trivial functional dependency (i.e., a dependency where the determinant is not a candidate key) is enforced by a candidate key. Hence, the tables are in BCNF.

### 5NF (Fifth Normal Form)

A table is in 5NF if and only if every join dependency in the table is implied by the candidate keys. 5NF is the highest level of normalization and ensures that the database schema is free of redundancies and inconsistencies. Hence, the tables are in 5NF.

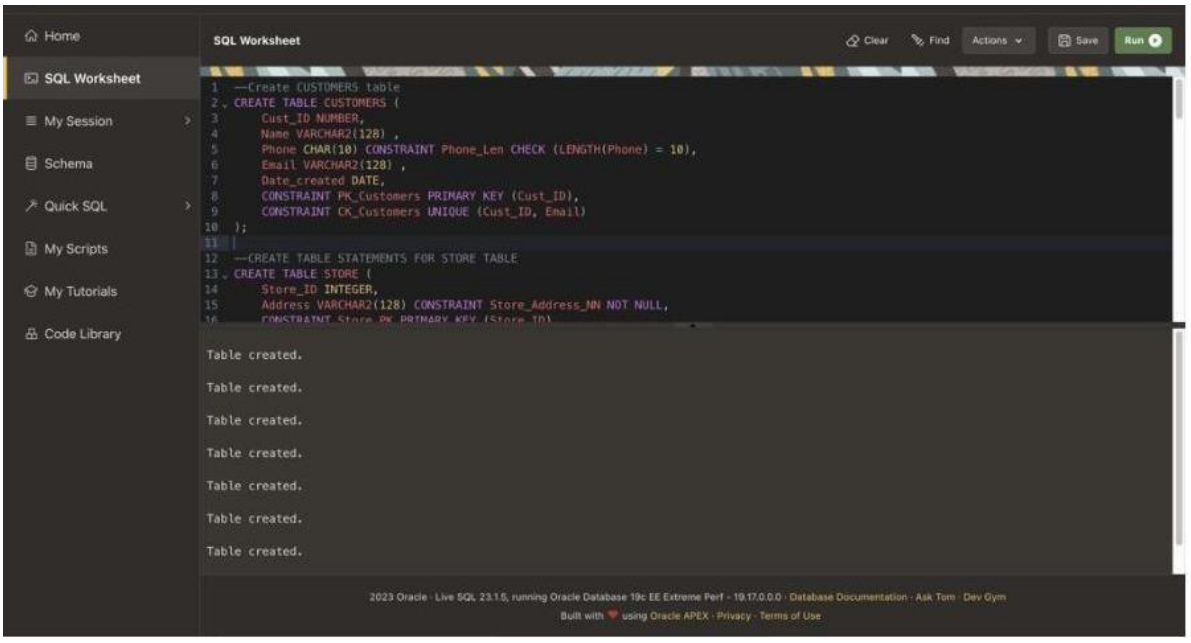
# SQL & PL/SQL COMMANDS

## SQL commands for creation of tables

```
--Create CUSTOMERS table
CREATE TABLE CUSTOMERS (
   Cust_ID  NUMBER,
   Name VARCHAR2(128) ,
   Phone CHAR(10) CONSTRAINT Phone_Len CHECK (LENGTH(Phone) = 10),
   Email VARCHAR2(128) ,
   Date_created DATE,
   CONSTRAINT PK_Customers PRIMARY KEY (Cust_ID),
   CONSTRAINT CK_Customers UNIQUE (Cust_ID, Email)
);

--CREATE TABLE STATEMENTS FOR STORE TABLE
CREATE TABLE STORE (
   Store_ID INTEGER,
```

```
    Address VARCHAR2(128) CONSTRAINT Store_Address_NN NOT NULL,
    CONSTRAINT Store_PK PRIMARY KEY (Store_ID),
    CONSTRAINT Store_UK UNIQUE (Address)
);

--Create EMPLOYEES table
CREATE TABLE EMPLOYEES (
    Emp_ID NUMBER,
    Name VARCHAR2(128) ,
    SSN CHAR(12) CONSTRAINT SSN_Len CHECK (LENGTH(SSN) = 12),
    Phone CHAR(10) CONSTRAINT EmpPhone_Len CHECK (LENGTH(Phone) = 10),
    Address VARCHAR2(256),
    Pay NUMBER(8,2),
    Store_ID NUMBER,
    Date_of_joining DATE,
    Date_of_leaving DATE,
    Email VARCHAR2(128),
    Password VARCHAR2(256),
    CONSTRAINT PK_Employees PRIMARY KEY (Emp_ID),
    CONSTRAINT CK_Employees UNIQUE (Emp_ID, SSN, Name, Email),
   CONSTRAINT FK_Employees_Store FOREIGN KEY (Store_ID) REFERENCES
STORE(Store_ID)
);

--Create CHECKOUT table
CREATE TABLE CHECKOUT (
    Checkout_ID NUMBER,
    Cust_ID NUMBER,
    Store_ID NUMBER,
    Tax NUMBER(8,2),
    Subtotal NUMBER(8,2),
    Emp_ID NUMBER,
    Date_of_order DATE,
    CONSTRAINT PK_Checkout PRIMARY KEY (Checkout_ID),
    CONSTRAINT FK_Checkout_Customer FOREIGN KEY (Cust_ID) REFERENCES
CUSTOMERS(Cust_ID),
    CONSTRAINT FK_Checkout_Store FOREIGN KEY (Store_ID) REFERENCES
STORE(Store_ID),
    CONSTRAINT FK_Checkout_Employee FOREIGN KEY (Emp_ID) REFERENCES
EMPLOYEES(Emp_ID),
    CONSTRAINT CK_Checkout UNIQUE (Checkout_ID, Cust_ID, Store_ID)
);

--Create ITEMS table
CREATE TABLE ITEMS (
    Items_ID NUMBER,
    Description VARCHAR2(128) ,
    Brand VARCHAR2(32) ,
    Cost NUMBER(8,2),
    Price NUMBER(8,2) CONSTRAINT Price_Positive CHECK (Price > 0),
    Weight NUMBER(8,2) CONSTRAINT Weight_Positive CHECK (Weight > 0),
    UPC NUMBER CONSTRAINT UPC_Positive CHECK (UPC > 0),
    CONSTRAINT PK_Items PRIMARY KEY (Items_ID)
);
```

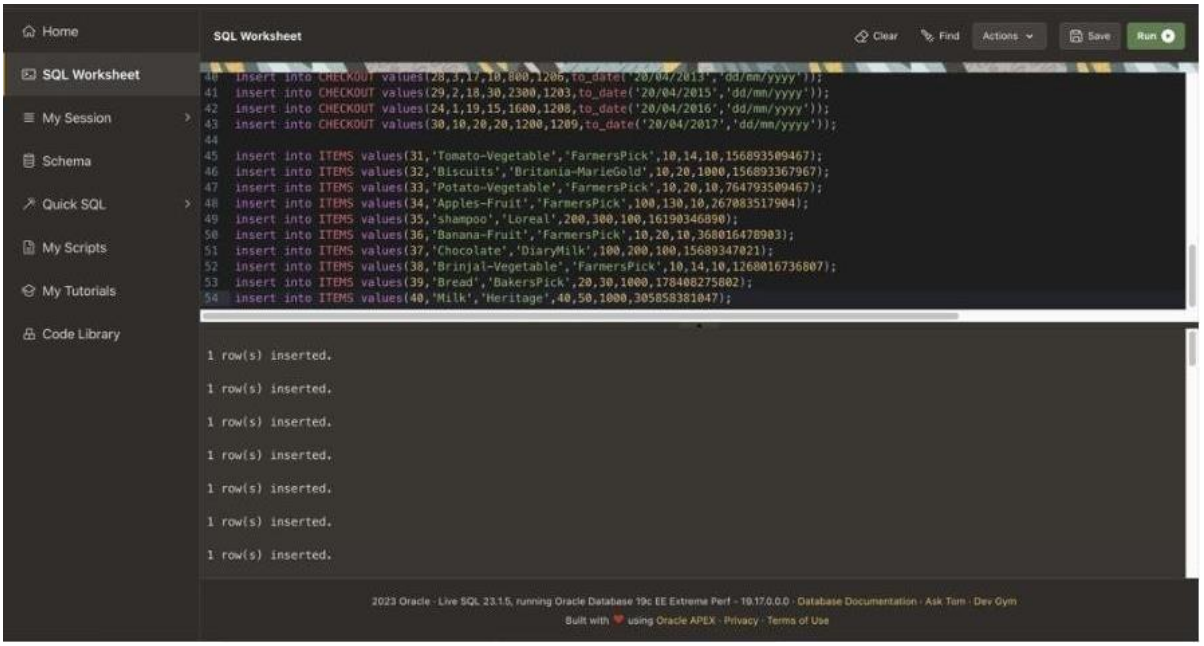
```
--Create CHECKOUTACTION table
CREATE TABLE CHECKOUTACTION (
   Checkout_ID NUMBER,
   Items_ID NUMBER,
   Quantity INTEGER,
   CONSTRAINT PK_CheckoutAction PRIMARY KEY (Checkout_ID, Items_ID),
   CONSTRAINT FK_CheckoutAction_Checkout FOREIGN KEY (Checkout_ID) REFERENCES
CHECKOUT(Checkout_ID),
   CONSTRAINT FK_CheckoutAction_Items FOREIGN KEY (Items_ID) REFERENCES
ITEMS(Items_ID)
);

--CREATE TABLE STATEMENTS FOR INVENTORY TABLE
CREATE TABLE INVENTORY (
   Items_ID INTEGER CONSTRAINT Inventory_ItemsID_NN NOT NULL REFERENCES
ITEMS(Items_ID),
   Store_ID INTEGER CONSTRAINT Inventory_StoreID_NN NOT NULL REFERENCES
STORE(Store_ID),
   Quantity INTEGER CONSTRAINT Inventory_Quantity_NN NOT NULL,
   CONSTRAINT Inventory_PK PRIMARY KEY (Items_ID, Store_ID)
);

--CREATE TABLE STATEMENTS FOR MANAGES_FOR TABLE
CREATE TABLE MANAGES_FOR (
   Emp_ID INTEGER CONSTRAINT ManagesFor_EmpID_NN NOT NULL REFERENCES
EMPLOYEES(Emp_ID),
   Store_ID INTEGER CONSTRAINT ManagesFor_StoreID_NN NOT NULL REFERENCES
STORE(Store_ID),
   Position VARCHAR2(32) CONSTRAINT ManagesFor_Position_NN NOT NULL,
   CONSTRAINT ManagesFor_PK PRIMARY KEY (Emp_ID, Store_ID)
);
```

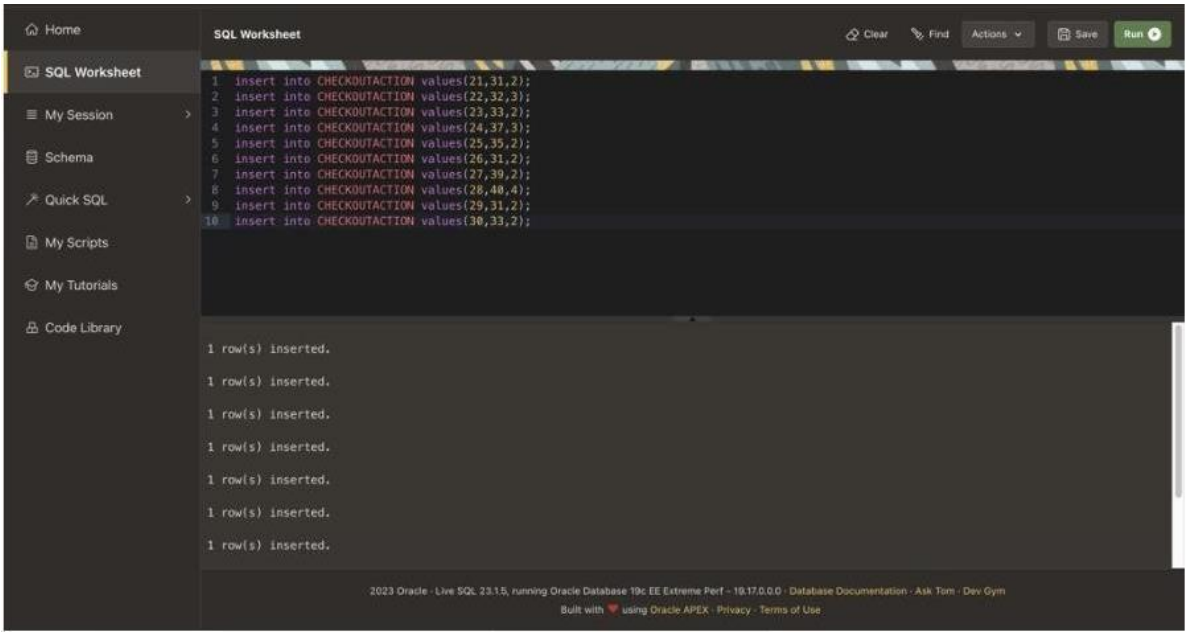


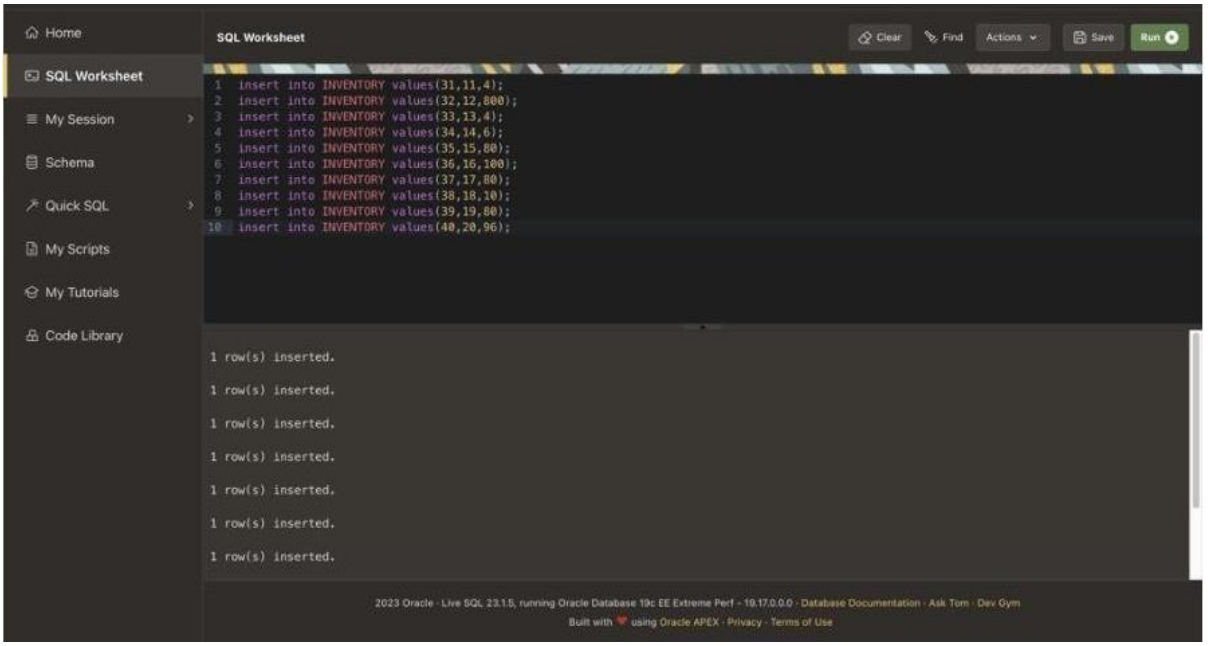
## Inserting values in tables

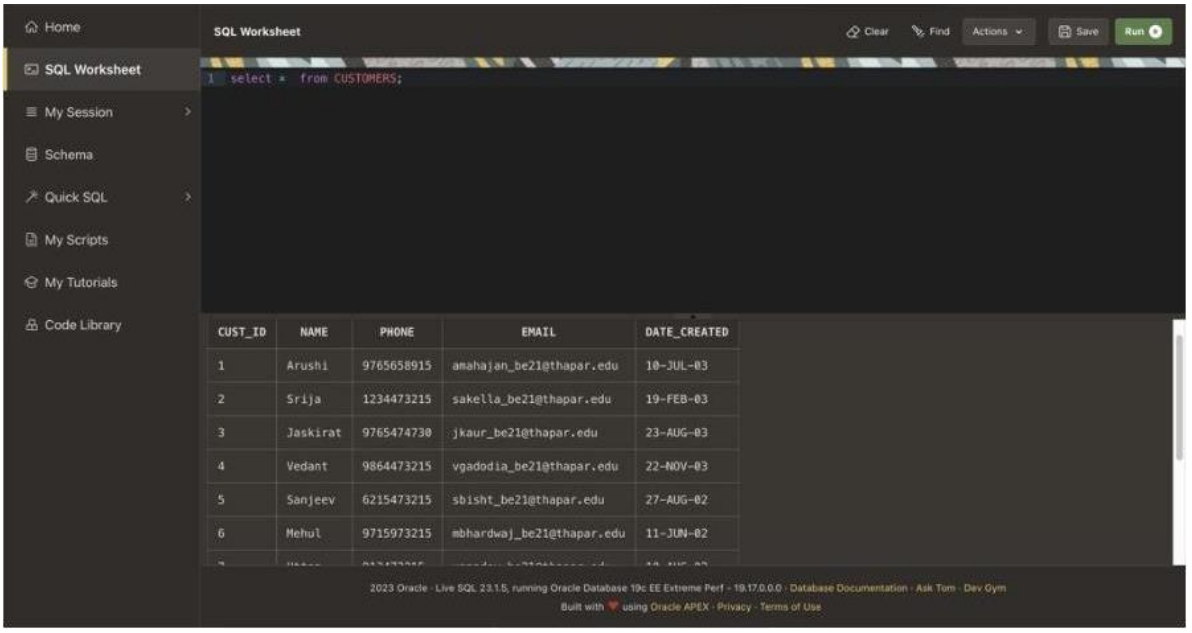
```
insert into CUSTOMERS
values(1,'Arushi',9765658915,'amahajan_be21@thapar.edu',to_date('10/07/2003','dd/mm/yyyy'));
insert into CUSTOMERS
values(2,'Srija',1234473215,'sakella_be21@thapar.edu',to_date('19/02/2003','dd/mm/yyyy'));
insert into CUSTOMERS
values(3,'Jaskirat',9765474730,'jkaur_be21@thapar.edu',to_date('23/08/2003','dd/mm/yyyy'));
insert into CUSTOMERS
values(4,'Vedant',9864473215,'vgadodia_be21@thapar.edu',to_date('22/11/2003','dd/mm/yyyy'));
insert into CUSTOMERS
values(5,'Sanjeev',6215473215,'sbisht_be21@thapar.edu',to_date('27/08/2002','dd/mm/yyyy'));
insert into CUSTOMERS
values(6,'Mehul',9715973215,'mbhardwaj_be21@thapar.edu',to_date('11/06/2002','dd/mm/yyyy'));
insert into CUSTOMERS
values(7,'Uttam',0913473215,'upandey_be21@thapar.edu',to_date('10/08/2002','dd/mm/yyyy'));
insert into CUSTOMERS
values(8,'Aditi',6189473215,'avij_be21@thapar.edu',to_date('15/12/2002','dd/mm/yyyy'));
insert into CUSTOMERS
values(9,'Svea',7645473215,'schawla_be21@thapar.edu',to_date('02/03/2003','dd/mm/yyyy'));
insert into CUSTOMERS
values(10,'Ishita',8745473215,'iarora_be21@thapar.edu',to_date('18/02/2004','dd/mm/yyyy'));

insert into STORE values(11,'Patiala,Leela Bhawan,Punjab,147004');
insert into STORE values(12,'Hyderabad,Gachbowli,Telangana,500032');
insert into STORE values(13,'Zeerakpur,Lohgarh,Punjab,140603');
insert into STORE values(14,'Akola,Rajeshwari Mandir,Maharashtra,444002');
insert into STORE values(15,'Chandigarh,sector 17,Punjab,160022');
insert into STORE values(16,'Chadigarh,sector 35,Punjab,160021');
insert into STORE values(17,'Lucknow,Gomti Nagar,UP,226010');
insert into STORE values(18,'Hyderabad,Malakajgiri,Telangana,500017');
insert into STORE values(19,'Agra,Shastri Nagar,UP,223007');
insert into STORE values(20,'Ghaziabad,Mandoli Saboli,UP,110093');

insert into
EMPLOYEES(Emp_ID,Name,SSN,Phone,Address,Pay,Store_ID,Date_of_Joining,Email,Password)v
alues(1200,'Samarth','928416732465','1234567897', 'Patiala', 20000,
11,to_date('19/05/2005','dd/mm/yyyy'),'skalra_be21@thapar.edu','S1234');
insert into
EMPLOYEES(Emp_ID,Name,SSN,Phone,Address,Pay,Store_ID,Date_of_Joining,Email,Password)v
alues(1201,'Sankalp','730648295601','5678567897', 'Akola', 18000,
14,to_date('19/06/2005','dd/mm/yyyy'),'sbajwa_be21@thapar.edu','S1235');
insert into
EMPLOYEES(Emp_ID,Name,SSN,Phone,Address,Pay,Store_ID,Date_of_Joining,Email,Password)v
alues(1202,'Arjun','524607934018','1234670297', 'Hyderabad', 16000,
12,to_date('19/07/2004','dd/mm/yyyy'),'akumar_be21@thapar.edu','A1234');
insert into
EMPLOYEES(Emp_ID,Name,SSN,Phone,Address,Pay,Store_ID,Date_of_Joining,Email,Password)v
alues(1203,'Ajay','970256483029','0243567897', 'Lucknow', 19000,
17,to_date('19/08/2003','dd/mm/yyyy'),'asingh_be21@thapar.edu','A2345');
insert into
EMPLOYEES(Emp_ID,Name,SSN,Phone,Address,Pay,Store_ID,Date_of_Joining,Email,Password)v
```

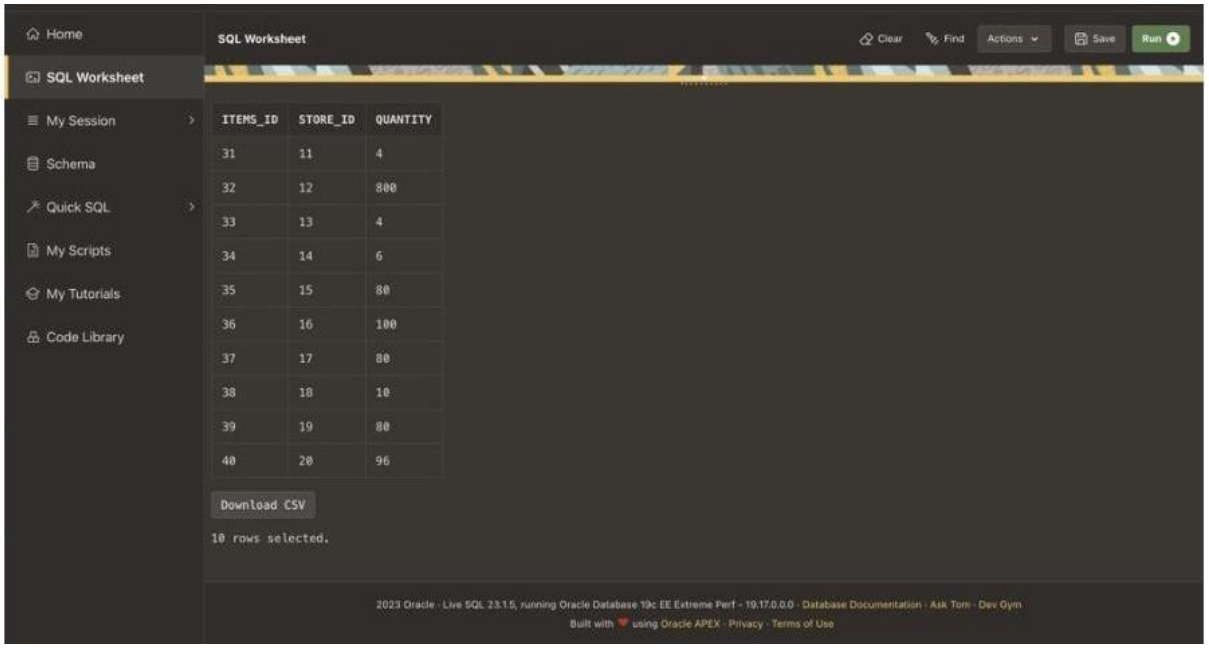
```
alues(1204,'Bharghav','086490234871','1234560123', 'Chandigarh', 20000,
15,to_date('19/09/2002','dd/mm/yyyy'),'bshukla_be21@thapar.edu','B5678');


insert into CHECKOUT values(21,9,11,20,1103,1200,to_date('20/04/2010','dd/mm/yyyy'));
insert into CHECKOUT values(22,8,12,30,1000,1201,to_date('20/06/2011','dd/mm/yyyy'));
insert into CHECKOUT values(23,7,13,10,2000,1200,to_date('20/07/2010','dd/mm/yyyy'));
insert into CHECKOUT values(25,6,14,11,1130,1202,to_date('20/08/2020','dd/mm/yyyy'));
insert into CHECKOUT values(26,5,15,29,3000,1204,to_date('20/03/2008','dd/mm/yyyy'));
insert into CHECKOUT values(27,4,16,20,3456,1203,to_date('20/04/2012','dd/mm/yyyy'));
insert into CHECKOUT values(28,3,17,10,800,1201,to_date('20/04/2013','dd/mm/yyyy'));
insert into CHECKOUT values(29,2,18,30,2300,1203,to_date('20/04/2015','dd/mm/yyyy'));
insert into CHECKOUT values(24,1,19,15,1600,1200,to_date('20/04/2016','dd/mm/yyyy'));
insert into CHECKOUT values(30,10,20,20,1200,1202,to_date('20/04/2017','dd/mm/yyyy'));

insert into ITEMS values(31,'Tomato-Vegetable','FarmersPick',10,14,10,156893509467);
insert into ITEMS values(32,'Biscuits','Britania-MarieGold',10,20,1000,156893367967);
insert into ITEMS values(33,'Potato-Vegetable','FarmersPick',10,20,10,764793509467);
insert into ITEMS values(34,'Apples-Fruit','FarmersPick',100,130,10,267083517904);
insert into ITEMS values(35,'shampoo','Loreal',200,300,100,16190346890);
insert into ITEMS values(36,'Banana-Fruit','FarmersPick',10,20,10,368016478903);
insert into ITEMS values(37,'Chocolate','DiaryMilk',100,200,100,15689347021);
insert into ITEMS values(38,'Brinjal-Vegetable','FarmersPick',10,14,10,1268016736807);
insert into ITEMS values(39,'Bread','BakersPick',20,30,1000,178408275802);
insert into ITEMS values(40,'Milk','Heritage',40,50,1000,305858381047);

insert into CHECKOUTACTION values(21,31,2);
insert into CHECKOUTACTION values(22,32,3);
insert into CHECKOUTACTION values(23,33,2);
insert into CHECKOUTACTION values(24,37,3);
insert into CHECKOUTACTION values(25,35,2);
insert into CHECKOUTACTION values(26,31,2);
insert into CHECKOUTACTION values(27,39,2);
insert into CHECKOUTACTION values(28,40,4);
insert into CHECKOUTACTION values(29,31,2);
insert into CHECKOUTACTION values(30,33,2);

insert into INVENTORY values(31,11,4);
insert into INVENTORY values(32,12,800);
insert into INVENTORY values(33,13,4);
insert into INVENTORY values(34,14,6);
insert into INVENTORY values(35,15,80);
insert into INVENTORY values(36,16,100);
insert into INVENTORY values(37,17,80);
insert into INVENTORY values(38,18,10);
insert into INVENTORY values(39,19,80);
insert into INVENTORY values(40,20,96);

insert into MANAGES_FOR values(1200,11,'Store Manager');
insert into MANAGES_FOR values(1201,12,'Store Manager');
insert into MANAGES_FOR values(1203,11,'Inventory Manager');
insert into MANAGES_FOR values(1202,13,'Store Manager');
insert into MANAGES_FOR values(1203,14,'Store Manager');
```


Home
SQL Worksheet
My Session
Schema
Quick SQL
My Scripts
My Tutorials
Code Library
SQL Worksheet
Clear
Find
Actions
Save
Run
insert into CHECKOUT values(28,3,17,10,800,1206,to_date('20/04/2013','dd/mm/yyyy'));
insert into CHECKOUT values(29,2,18,30,2300,1203,to_date('20/04/2015','dd/mm/yyyy'));
insert into CHECKOUT values(24,1,19,15,1600,1208,to_date('20/04/2016','dd/mm/yyyy'));
insert into CHECKOUT values(30,10,20,20,1200,1209,to_date('20/04/2017','dd/mm/yyyy'));
insert into ITEMS values(31,'Tomato-Vegetable','FarmersPick',10,14,10,156893509467);
insert into ITEMS values(32,'Biscuits','Britania-MarieGold',10,20,1000,156893367967);
insert into ITEMS values(33,'Potato-Vegetable','FarmersPick',10,20,10,764793509467);
insert into ITEMS values(34,'Apples-Fruit','FarmersPick',100,130,10,267083517904);
insert into ITEMS values(35,'shampoo','Loreal',200,300,100,16190346890);
insert into ITEMS values(36,'Banana-Fruit','FarmersPick',10,20,10,368016478903);
insert into ITEMS values(37,'Chocolate','DiaryMilk',100,200,100,15689347021);
insert into ITEMS values(38,'Brinjal-Vegetable','FarmersPick',10,14,10,1268016736807);
insert into ITEMS values(39,'Bread','BakersPick',20,30,1000,178408275802);
insert into ITEMS values(40,'Milk','Heritage',40,50,1000,305858381047);
1 row(s) inserted.
1 row(s) inserted.
1 row(s) inserted.
1 row(s) inserted.
1 row(s) inserted.
1 row(s) inserted.
1 row(s) inserted.
2023 Oracle · Live SQL 23.1.5, running Oracle Database 19c EE Extreme Perf - 19.17.0.0.0 · Database Documentation · Ask Tom · Dev Gym
Built with using Oracle APEX · Privacy · Terms of Use

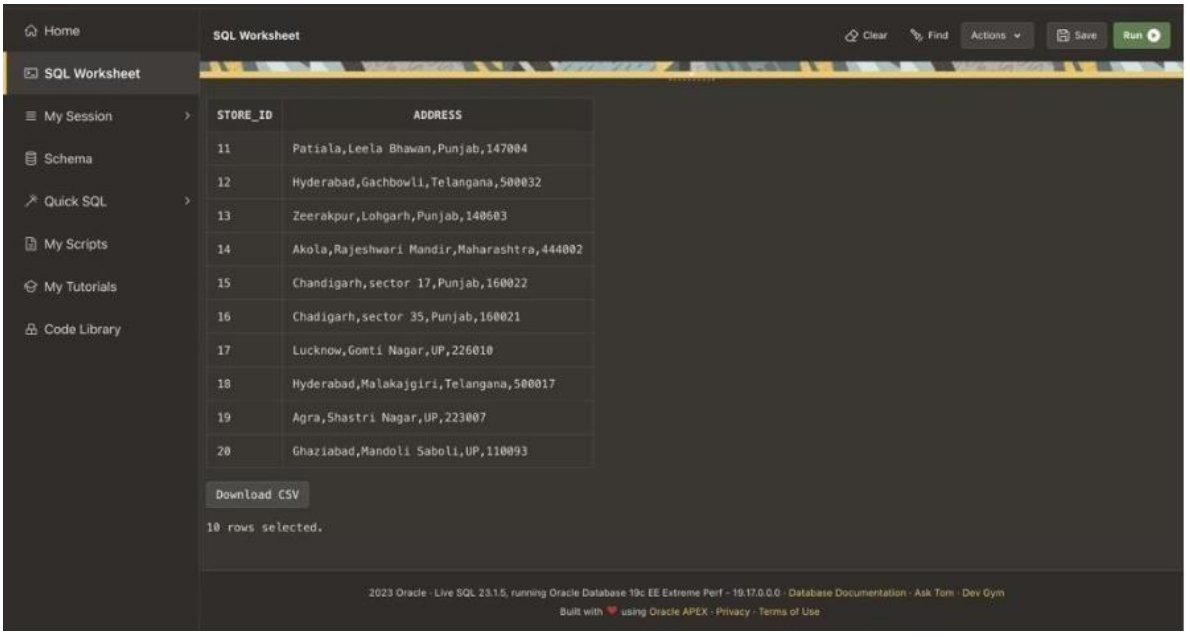

Home
SQL Worksheet
My Session
Schema
Quick SQL
My Scripts
My Tutorials
Code Library
SQL Worksheet
Clear
Find
Actions
Save
Run
insert into CHECKOUTACTION values(21,31,2);
insert into CHECKOUTACTION values(22,32,3);
insert into CHECKOUTACTION values(23,33,2);
insert into CHECKOUTACTION values(24,37,3);
insert into CHECKOUTACTION values(25,35,2);
insert into CHECKOUTACTION values(26,31,2);
insert into CHECKOUTACTION values(27,39,2);
insert into CHECKOUTACTION values(28,40,4);
insert into CHECKOUTACTION values(29,31,2);
insert into CHECKOUTACTION values(30,33,2);
1 row(s) inserted.
1 row(s) inserted.
1 row(s) inserted.
1 row(s) inserted.
1 row(s) inserted.
1 row(s) inserted.
1 row(s) inserted.
2023 Oracle · Live SQL 23.1.5, running Oracle Database 19c EE Extreme Perf - 19.17.0.0.0 · Database Documentation · Ask Tom · Dev Gym
Built with using Oracle APEX · Privacy · Terms of Use


Home
SQL Worksheet
My Session
Schema
Quick SQL
My Scripts
My Tutorials
Code Library
SQL Worksheet
Clear
Find
Actions
Save
Run
insert into INVENTORY values(31,11,4);
insert into INVENTORY values(32,12,800);
insert into INVENTORY values(33,13,4);
insert into INVENTORY values(34,14,6);
insert into INVENTORY values(35,15,80);
insert into INVENTORY values(36,16,100);
insert into INVENTORY values(37,17,80);
insert into INVENTORY values(38,18,10);
insert into INVENTORY values(39,19,80);
insert into INVENTORY values(40,20,96);
1 row(s) inserted.
1 row(s) inserted.
1 row(s) inserted.
1 row(s) inserted.
1 row(s) inserted.
1 row(s) inserted.
1 row(s) inserted.
2023 Oracle · Live SQL 23.1.5, running Oracle Database 19c EE Extreme Perf - 19.17.0.0.0 · Database Documentation · Ask Tom · Dev Gym
Built with using Oracle APEX · Privacy · Terms of Use

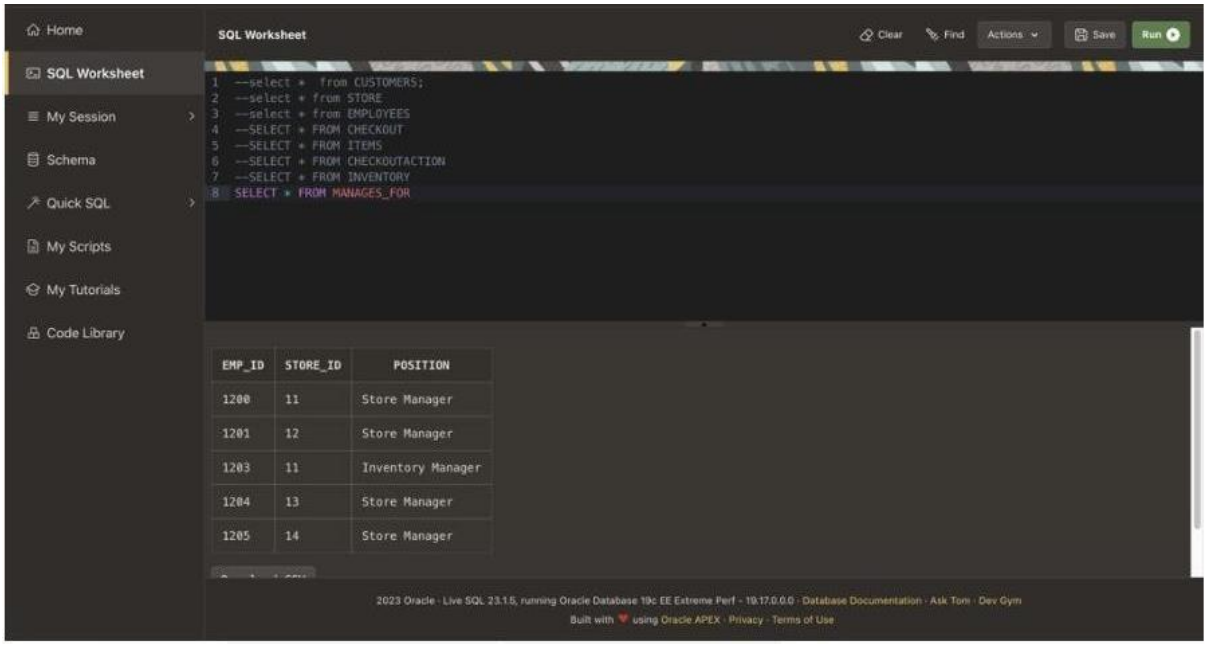

Home
SQL Worksheet
My Session
Schema
Quick SQL
My Scripts
My Tutorials
Code Library
SQL Worksheet
Clear
Find
Actions
Save
Run
select * from CUSTOMERS;
CUST_ID NAME PHONE EMAIL DATE_CREATED
1 Arushi 9765658915 amahajan_be21@thapar.edu 10-JUL-03
2 Srija 1234473215 sakella_be21@thapar.edu 19-FEB-03
3 Jaskirat 9765474730 jkaur_be21@thapar.edu 23-AUG-03
4 Vedant 9864473215 vgadodia_be21@thapar.edu 22-NOV-03
5 Sanjeev 6215473215 sbisht_be21@thapar.edu 27-AUG-02
6 Mehul 9715973215 mbhardwaj_be21@thapar.edu 11-JUN-02
2023 Oracle · Live SQL 23.1.5, running Oracle Database 19c EE Extreme Perf - 19.17.0.0.0 · Database Documentation · Ask Tom · Dev Gym
Built with using Oracle APEX · Privacy · Terms of Use

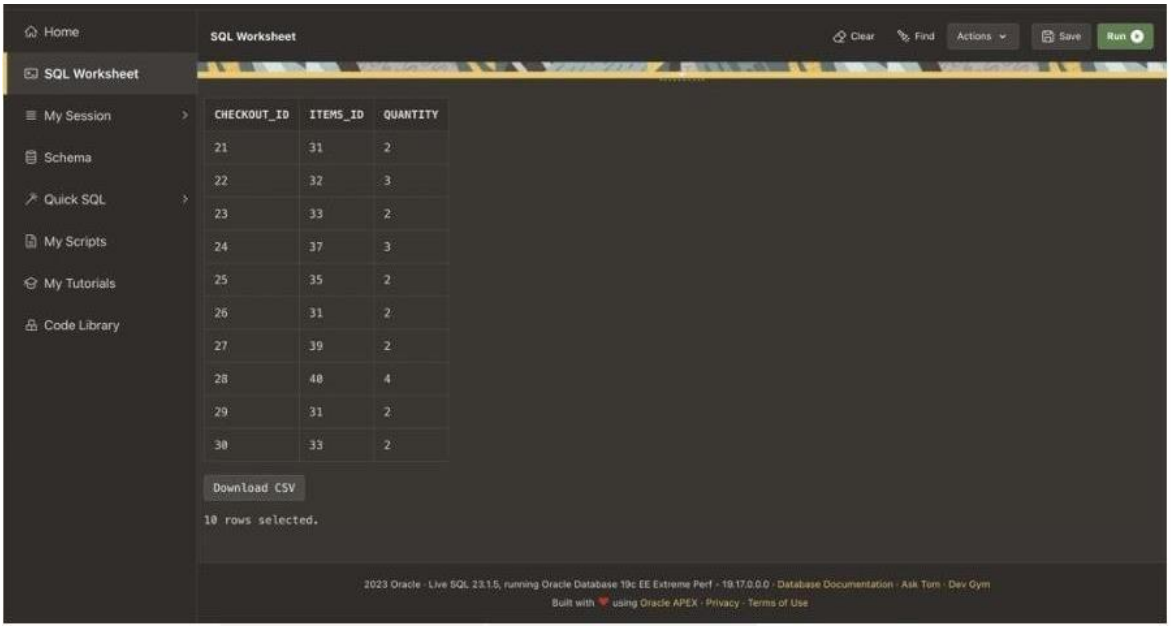


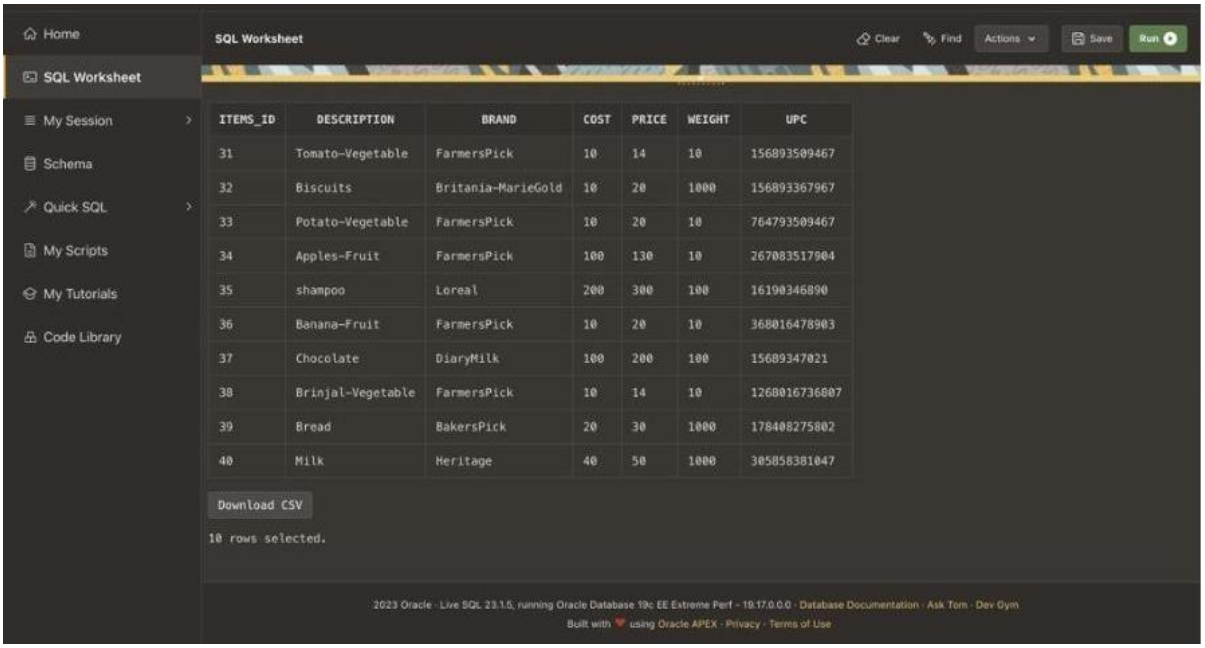

| ITEMS_ID | STORE_ID | QUANTITY |
|---|---|---|
| 31 | 11 | 4 |
| 32 | 12 | 800 |
| 33 | 13 | 4 |
| 34 | 14 | 6 |
| 35 | 15 | 80 |
| 36 | 16 | 100 |
| 37 | 17 | 80 |
| 38 | 18 | 10 |
| 39 | 19 | 80 |
| 40 | 20 | 96 |




| STORE_ID | ADDRESS |
|---|---|
| 11 | Patiala,Leela Bhawan,Punjab,147004 |
| 12 | Hyderabad,Gachbowli,Telangana,500032 |
| 13 | Zeerakpur,Lohgarh,Punjab,140603 |
| 14 | Akola,Rajeshwari Mandir,Maharashtra,444002 |
| 15 | Chandigarh,sector 17,Punjab,160022 |
| 16 | Chadigarh,sector 35,Punjab,160021 |
| 17 | Lucknow,Gomti Nagar,UP,226010 |
| 18 | Hyderabad,Malakajgiri,Telangana,500017 |
| 19 | Agra,Shastri Nagar,UP,223007 |
| 20 | Ghaziabad,Mandoli Saboli,UP,110093 |


Home
SQL Worksheet
My Session
Schema
Quick SQL
My Scripts
My Tutorials
Code Library
SQL Worksheet
Clear
Find
Actions
Save
Run
--select * from CUSTOMERS;
--select * from STORE
--select * from EMPLOYEES
--SELECT * FROM CHECKOUT
--SELECT * FROM ITEMS
--SELECT * FROM CHECKOUTACTION
--SELECT * FROM INVENTORY
SELECT * FROM MANAGES_FOR
EMP_ID STORE_ID POSITION
1200 11 Store Manager
1201 12 Store Manager
1203 11 Inventory Manager
1204 13 Store Manager
1205 14 Store Manager
2023 Oracle · Live SQL 23.1.5, running Oracle Database 19c EE Extreme Perf - 19.17.0.0.0 · Database Documentation · Ask Tom · Dev Gym
Built with using Oracle APEX · Privacy · Terms of Use


Home
SQL Worksheet
My Session
Schema
Quick SQL
My Scripts
My Tutorials
Code Library
SQL Worksheet
Clear
Find
Actions
Save
Run
CHECKOUT_ID ITEMS_ID QUANTITY
21 31 2
22 32 3
23 33 2
24 37 3
25 35 2
26 31 2
27 39 2
28 40 4
29 31 2
30 33 2
Download CSV
10 rows selected.
2023 Oracle · Live SQL 23.1.5, running Oracle Database 19c EE Extreme Perf - 19.17.0.0.0 · Database Documentation · Ask Tom · Dev Gym
Built with using Oracle APEX · Privacy · Terms of Use

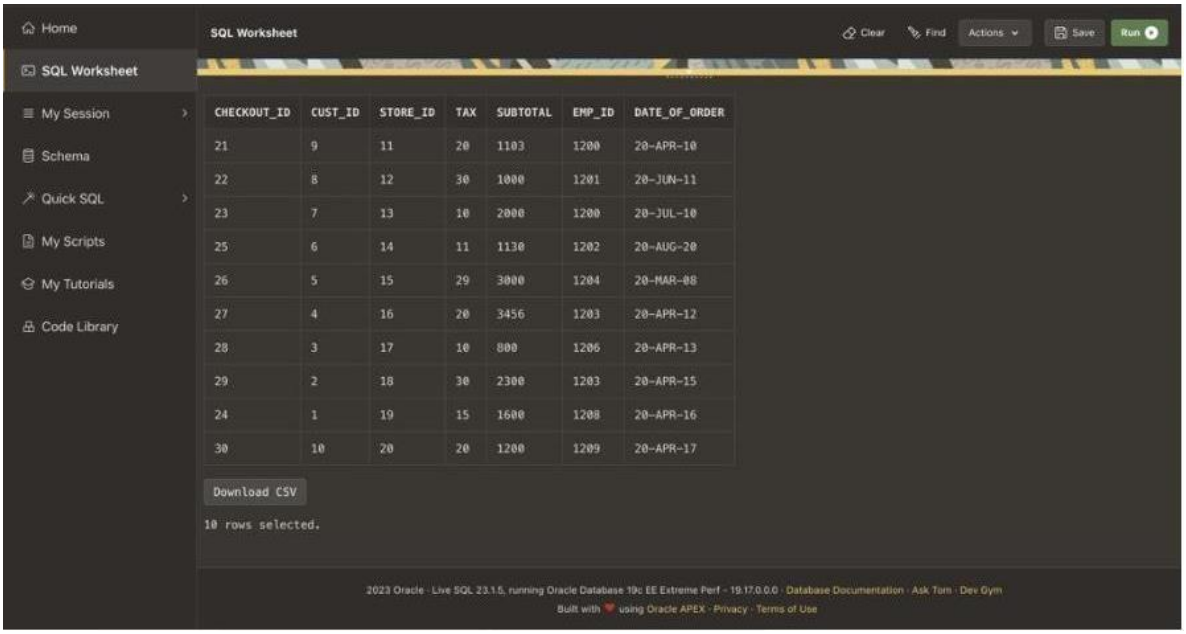


SQL Worksheet

| ITEMS_ID | DESCRIPTION | BRAND | COST | PRICE | WEIGHT | UPC |
|---|---|---|---|---|---|---|
| 31 | Tomato-Vegetable | FarmersPick | 10 | 14 | 10 | 156893509467 |
| 32 | Biscuits | Britania-MarieGold | 10 | 20 | 1000 | 156893367967 |
| 33 | Potato-Vegetable | FarmersPick | 10 | 20 | 10 | 764793509467 |
| 34 | Apples-Fruit | FarmersPick | 100 | 130 | 10 | 267083517904 |
| 35 | shampoo | Loreal | 200 | 300 | 100 | 16190346890 |
| 36 | Banana-Fruit | FarmersPick | 10 | 20 | 10 | 368016478903 |
| 37 | Chocolate | DiaryMilk | 100 | 200 | 100 | 15689347021 |
| 38 | Brinjal-Vegetable | FarmersPick | 10 | 14 | 10 | 1268016736807 |
| 39 | Bread | BakersPick | 20 | 30 | 1000 | 178408275802 |
| 40 | Milk | Heritage | 40 | 50 | 1000 | 305858381047 |

Download CSV

10 rows selected.



SQL Worksheet

| CHECKOUT_ID | CUST_ID | STORE_ID | TAX | SUBTOTAL | EMP_ID | DATE_OF_ORDER |
|---|---|---|---|---|---|---|
| 21 | 9 | 11 | 20 | 1103 | 1200 | 20-APR-10 |
| 22 | 8 | 12 | 30 | 1000 | 1201 | 20-JUN-11 |
| 23 | 7 | 13 | 10 | 2000 | 1200 | 20-JUL-10 |
| 25 | 6 | 14 | 11 | 1130 | 1202 | 20-AUG-20 |
| 26 | 5 | 15 | 29 | 3000 | 1204 | 20-MAR-08 |
| 27 | 4 | 16 | 20 | 3456 | 1203 | 20-APR-12 |
| 28 | 3 | 17 | 10 | 800 | 1206 | 20-APR-13 |
| 29 | 2 | 18 | 30 | 2300 | 1203 | 20-APR-15 |
| 24 | 1 | 19 | 15 | 1600 | 1208 | 20-APR-16 |
| 30 | 10 | 20 | 20 | 1200 | 1209 | 20-APR-17 |

Download CSV

10 rows selected.

| EMP_ID | STORE_ID | POSITION |
|---|---|---|
| 1200 | 11 | Store Manager |
| 1201 | 12 | Store Manager |
| 1203 | 11 | Inventory Manager |
| 1202 | 13 | Store Manager |
| 1203 | 14 | Store Manager |

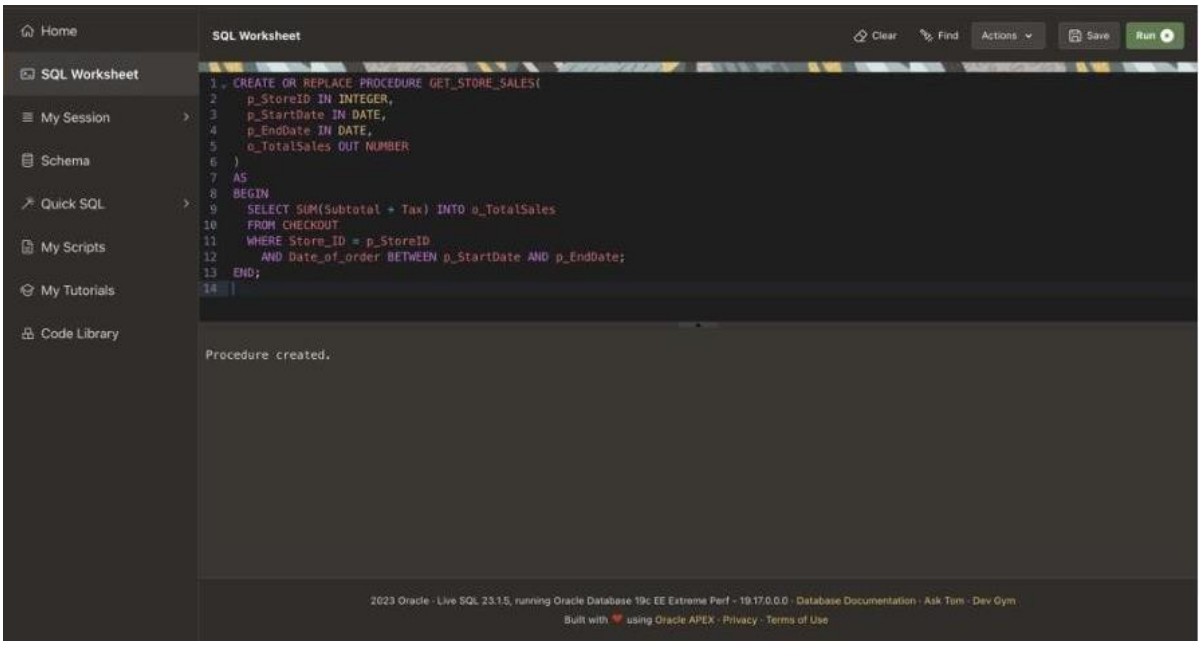
| CHECKOUT_ID | CUST_ID | STORE_ID | TAX | SUBTOTAL | EMP_ID | DATE_OF_ORDER |
|---|---|---|---|---|---|---|
| 21 | 9 | 11 | 20 | 1103 | 1200 | 20-APR-10 |
| 22 | 8 | 12 | 30 | 1000 | 1201 | 20-JUN-11 |
| 23 | 7 | 13 | 10 | 2000 | 1200 | 20-JUL-10 |
| 25 | 6 | 14 | 11 | 1130 | 1202 | 20-AUG-20 |
| 26 | 5 | 15 | 29 | 3000 | 1204 | 20-MAR-08 |
| 27 | 4 | 16 | 20 | 3456 | 1203 | 20-APR-12 |

| EMP_ID | NAME | SSN | PHONE | ADDRESS | PAY | STORE_ID | DATE_OF_JOINING | DATE_OF_LEAVING | EMAIL | PASSWORD |
|---|---|---|---|---|---|---|---|---|---|---|
| 1200 | Samarth | 928416732465 | 1234567897 | Patiala | 20000 | 11 | 19-MAY-05 | - | skalra_be21@thapar.edu | S1234 |
| 1201 | Sankalp | 730648295601 | 5678567897 | Akola | 18000 | 14 | 19-JUN-05 | - | sbajwa_be21@thapar.edu | S1235 |
| 1202 | Arjun | 524607934018 | 1234670297 | Hyderabad | 16000 | 12 | 19-JUL-04 | - | akumar_be21@thapar.edu | A1234 |
| 1203 | Ajay | 970256483029 | 0243567897 | Lucknow | 19000 | 17 | 19-AUG-03 | - | asingh_be21@thapar.edu | A2345 |
| 1204 | Bharghav | 086490234871 | 1234560123 | Chandigarh | 20000 | 15 | 19-SEP-02 | - | bshukla_be21@thapar.edu | B5678 |

# PL/SQL commands

**Stored procedure in Oracle that retrieves the total sales for a given store and date range:**

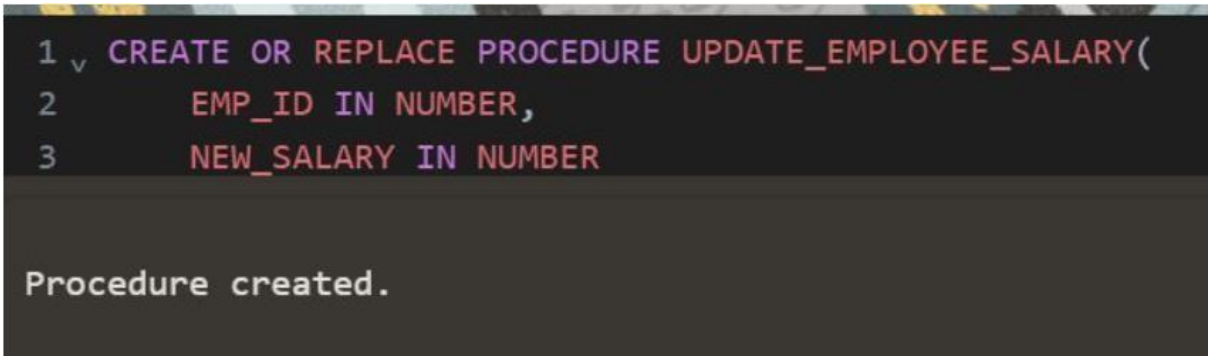
```
CREATE OR REPLACE PROCEDURE GET_STORE_SALES(
  p_StoreID IN INTEGER,
  p_StartDate IN DATE,
  p_EndDate IN DATE,
  o_TotalSales OUT NUMBER
)
AS
BEGIN
  SELECT SUM(Subtotal + Tax) INTO o_TotalSales
  FROM CHECKOUT
  WHERE Store_ID = p_StoreID
    AND Date_of_order BETWEEN p_StartDate AND p_EndDate;
END;
```




**Stored procedure that retrieves the updated employee salary based on employee ID:**

```
CREATE OR REPLACE PROCEDURE UPDATE_EMPLOYEE_SALARY(
   EMP_ID IN NUMBER,
   NEW_SALARY IN NUMBER
```

```
)
IS
BEGIN
   UPDATE EMPLOYEES
   SET Pay = NEW_SALARY
   WHERE Emp_ID = EMP_ID;

   DBMS_OUTPUT.PUT_LINE('Salary updated successfully.');
EXCEPTION
   WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
END;
```




**Stored procedure that add item to inventory:**

```
CREATE OR REPLACE PROCEDURE ADD_ITEM_TO_INVENTORY(
   STORE_ID IN NUMBER,
   ITEM_ID IN NUMBER,
   QUANTITY IN NUMBER
)
IS
BEGIN
   INSERT INTO INVENTORY (Store_ID, Items_ID, Quantity)
   VALUES (STORE_ID, ITEM_ID, QUANTITY);

   DBMS_OUTPUT.PUT_LINE('Item added to inventory successfully.');
EXCEPTION
   WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
END;
```

```
BEGIN
    ADD_ITEM_TO_INVENTORY(12, 38, 45);
END;
```

```
Statement processed.
Item added to inventory successfully.
```

**Trigger to update date_created when a new customer is added**

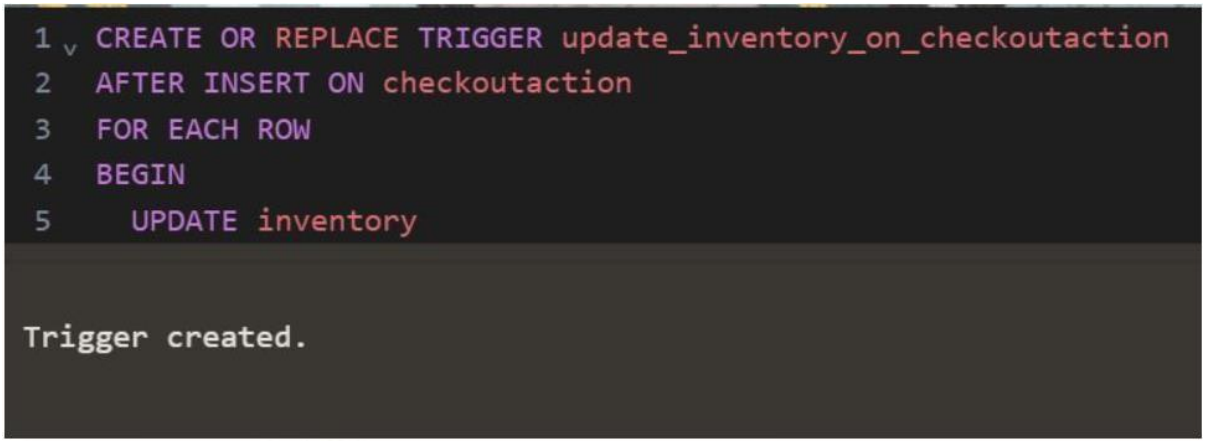
```
CREATE OR REPLACE TRIGGER checkout_trigger
AFTER INSERT ON checkout
FOR EACH ROW
BEGIN
  UPDATE customers
  SET date_created = SYSDATE
WHERE cust_id = :new.cust_id;
END;
```

```
CREATE OR REPLACE TRIGGER checkout_trigger
AFTER INSERT ON checkout
FOR EACH ROW
BEGIN
```

```
Trigger created.
```

### Trigger to update inventory whenever checkout action takes place

```
CREATE OR REPLACE TRIGGER update_inventory_on_checkoutaction
AFTER INSERT ON checkoutaction
FOR EACH ROW
BEGIN
  UPDATE inventory
  SET quantity = quantity - :NEW.quantity
WHERE items_id = :NEW.items_id;
END;
```




### Retrieve all customers who have placed an order in the last 30 days and their total amount spent

```
DECLARE
  v_date DATE := SYSDATE - 30; -- retrieve orders from the last 30 days
  v_total NUMBER;

  CURSOR customer_orders IS
    SELECT c.name, c.phone, c.email, SUM(ca.quantity * i.price) AS total_spent
    FROM customers c
    JOIN checkout co ON c.cust_id = co.cust_id
    JOIN checkoutaction ca ON co.checkout_id = ca.checkout_id
    JOIN items i ON ca.items_id = i.items_id
    WHERE co.date_of_order >= v_date
    GROUP BY c.name, c.phone, c.email;

BEGIN
  FOR cust_order IN customer_orders LOOP
    v_total := cust_order.total_spent;
    DBMS_OUTPUT.PUT_LINE(cust_order.name || ', ' || cust_order.phone || ', ' || cust_order.email || ', '
|| v_total);
END LOOP;
END;
```

```
DECLARE
    v_date DATE := SYSDATE - 30; -- retrieve orders from the last 30 days
    v_total NUMBER;

    CURSOR customer_orders IS
      SELECT c_name, c_phone, c_email, SUM(ca_quantity * i_price) AS total_spent
```

```
Statement processed.
```

**Procedure to retrieve total revenue from checkout**

```
CREATE OR REPLACE PROCEDURE GetCheckoutTotals(checkout_id IN NUMBER)
IS
   l_cost NUMBER(8,2);
   l_revenue NUMBER(8,2);
   CURSOR c_items IS
      SELECT i.Cost, i.Price, ca.Quantity
      FROM CHECKOUTACTION ca
      JOIN ITEMS i ON i.Items_ID = ca.Items_ID
      WHERE ca.Checkout_ID = checkout_id;

BEGIN
   l_cost := 0;
   l_revenue := 0;

   FOR item IN c_items LOOP
      l_cost := l_cost + (item.Cost * item.Quantity);
      l_revenue := l_revenue + (item.Price * item.Quantity);
   END LOOP;

   DBMS_OUTPUT.PUT_LINE('Total Cost: $' || l_cost);
   DBMS_OUTPUT.PUT_LINE('Total Revenue: $' || l_revenue);
END;
```

```
CREATE OR REPLACE PROCEDURE GetCheckoutTotalsS(checkout_id IN NUMBER)
IS
    l_cost NUMBER(8,2);
    l_revenue NUMBER(8,2);
    CURSOR c_items IS
```

```
Procedure created.
```

# CONCLUSION

The system provides features such as inventory management, employee management, customer management, and checkout management, making it easier for store owners to manage their stores. The system has been developed using SQL, a popular relational database management system, ensuring that the data is well-organized, secure, and easily accessible. With the use of the database, the store can keep track of its inventory, sales, and employee management. The system is expected to provide a platform for customers to manage their accounts, purchase items, and receive receipts. Overall, the project provides an efficient and cost-effective solution for grocery store management.

# REFERENCES


- https://www.slideshare.net
- https://docs.oracle.com
- https://www.tutorialspoint.com
- https://www.javatpoint.com