



Team 6.

- Chirag Padsala
- Harshal Saptarshi
- Suyog Shah

FINAL REPORT

TARGET ENTERPRISE (PTY) LTD.

Table of Contents

1. Introduction.....	3
2.Business Problem.....	3
3. Scope and focus of the Database Solutions	5
4. Parenthetical method of the Final Solution.....	6
5. Final ERD Illustration.....	9
6. Description of Changes.....	10
7.List of Queries, Stored Procedures and Triggers.....	13
8. Discuss Future Scope.....	27
9. Web App Prototype.....	27
10. Conclusion.....	35
11. Appendices.....	36

Table of Figures

5.1 Final ERD Diagram.....	9
7.1 Finding Order details	14
7.2.1 Itemized list of payment for credit card.....	15
7.2.2 Itemized list of payment for postal Order/cheque Number.....	16
7.3 Finding Shipment details	17
7.4.1 Trigger to insert data into “Audit_Trail” Table.....	18
7.4.2 Audit Trail entries.....	19
7.5 Updating orders for insufficient stock.	20
7.6 Refund of cancelled orders	21
7.7 Finding distinct mailing list	22
7.8.1 Finding bestselling item	23
7.8.2 Finding bestselling item by quantity sold	24
7.9 Print upcoming order.....	25
7.10 Shipping Details.....	25
7.11 Update order status automatically	26
9.1 Home page.....	27
9.2 Add order page.....	28
9.3.1 Shipping info.....	29
9.3.2 Shipment info.....	30
9.4.1 Bestselling item info.....	31
9.4.2 Inventory info.....	32
9.5.1 Audit Trail.....	33
9.5.2 Customer Info.....	34

1. Introduction

Target Enterprises PTY Ltd, a mail order company, needs to build a robust computerized order management system. The company needs to eliminate all paperwork by paving the way for an effective system. It needs to manage the customer database in an effective manner by observing consumer behaviour, purchase history and other related factors which can help the company improve sales. Eliminating duplicity and improving customer database form the crux of the company's issues which need to be addressed. The company is divided into 5 different verticals which are Marketing, Accounting, Order Processing, Dispatch and Purchasing. With the additional information provided in part 2, there were many additional functionalities that needs to be added.

2. Business Problem

Target Enterprise's business problems vary in complexity and urgency; however, they must be addressed to sustain profitability in short and long term. The issues mentioned are summarized below.

Business Problems: Combining Phase 1 and Phase 2 requirements

1. Customer database system- Target Enterprises (Pty) Ltd does not have a proper database system to maintain their customer list which could prove detrimental to their expansion plans.

- Issue #1- Customer's tastes and purchase history which the company doesn't have in their current manual system is considered as one of the issue. So, an automated system is required to fill the gap of the issue.

- Issue #2- This process is intended to handle changes to the database when a customer informs the Company that he or she has moved, or in the case where some other vital information about the customer has changed. This will help ensure against duplicate entries for a customer in the system.

2. Inefficient computing system - The computing systems used by the company's employees are not that efficient due to following issues:

- Issue #1- Accounting package is used only on one standalone computer which creates a risk of database stored on that computer. If the PC stops working due to some problems, then whole data about accounting package is lost which creates a big loophole in the company's data storing department.

- Issue #2- In the company, the ops manager is responsible for both ordering new stock, as well as handling incoming customer orders and dispatching them and does not have computer is also a big issue because no track record can be traced on a computer due to lack of resources.

- Issue #3- The marketing manager has an old PC with small dBase 3 system to maintain which is considered as an issue. In future if there is rapid growth in number of customers, the marketing manager would surely need a new system that can store more data compared to the old one.

3. Marketing Department - This department holds the mailing list entries for the old customers that purchased something from the company. The issues related to this are listed below:

- Issue #1- The marketing department are having lots of duplicate mailing list entries that is considered as an issue and to address that issue a new functionality should be created to address that issue.

- Issue #2- The department are having an issue in developing a report that shows best-selling items by value, or by number of units sold. The company needs to work on this issue to keep track of number of units sold as well as best selling items.

- Issue #3- The department are having trouble in developing the bestselling product by value, so the functionality should be added to solve the issue.

4. Ordering System- Ordering system consist of several steps such as order processing, order dispatch, purchasing and many other things. Issues related to ordering system is listed below:

- Issue #1- Ordering system lacks on functionality to track order status. To do so a proper functionality needs to be developed.

- Issue #2- The system lacks to validate the new order by keeping update of stock levels and stock items. Hence a proper functionality is needed to validate the new order by keeping update of stock.

- Issue #3- There may be chances of other issues in the order form. Need to scrutinize the order form properly and fix the issues if required.

- Issue #4- The system does not have the capability of a full audit transaction. A new functionality is added to solve the issue.

· Issue #5 – The current system lacks the functionality in optimizing picking and packing processes that needs to be added.

5. Accounting Department: - This department keeps track for banking incoming money and preparing reports about the financial position of the company.

Issue #1 – The current system doesn't provide the functionality for printed itemized lists of payments for this department. So, this functionality needs to be added to meet the requirements given.

3. Scope and Focus of the Database Solution

Objective: To implement a robust database system to address the immediate need of Target Enterprise, by concentrating on the marketing, ordering and dispatch processes, the purchase of new stock, and the accounting functions.

Deliverables: To complete the implementation of the database system timely and considering all the aspects, the following deliverable will be required:

1. Automate the recording and checking of incoming orders to improve the marketing function.
2. Capability to track the progress of order at any stage.
3. Printed itemized lists of payments for Accounting Department.
4. Optimize picking and packing processes.
5. Validation of new orders.
6. Full transaction auditability.
7. Report that shows the best-selling items by value, or by number of units sold.
8. Processing refund by either cancelling the order or by returning the product is added to the database design.

Milestones: It is necessary that automated system should be implemented quickly to solve the issues.

The milestones are:

- A. Creating a prototype of web application.
- B. Layout of home page with navigation to other pages.

C. Proper database system to solve the initial internal issues.

Limitations and Exclusions: The new order management system must be adaptable by the employees. As employees are not used to computer systems it will be required to train them and make them familiar with the systems.

Focus of the Database Solution: The focus of the database system, is to make a robust order management system that eliminates duplicate mailing list entries and support the accounting and marketing department functionality.

4. Parenthetical method of the Final Solution

1. **Customers (First_Name, Last_Name, Address, Shipping_Address, Zipcode, Mobile_Number, Email_Address)**

- This table contains the entry of every customer; this table will help the marketing department to avoid duplicate entries of a single customer.

2. **Orders (Order_ID, Customer_ID, Order_Mode_ID, Payment_Mode_ID, Status_ID, Delivery_Mode_ID, Order_Date, Total_Quantity_Ordered, isCanceled)**

- This table contains attributes of the order form which is filled by every customer. This is the main table of the ordering system, which contains foreign keys of other tables in the database.

3. **Shipping (Shipping_ID, Order_ID, Shipping_Date, Total_Quantity_Shipped, Shipping_Address)**

- This table stores the brief shipping details of items ordered by customers. Also it contains foreign key of Order_ID, which helps in retrieving the order detail as well as shipping details. It stores parameters like total quantity shipped, shipping date and shipping address.

4. **Shipping_Details (Shipping_Details_ID, Shipping_ID, Order_ID, Item_ID, Quantity_Ordered, Quantity_Shipped)**

- This table stores the shipping details of the customer and also the item details ordered by the customer. The main difference between the shipping and shipping_details table is in shipping_details table there is a foreign key of Item_ID which helps in retrieving the data of items from the inventory.

5. **CreditCard_Info (CreditCard_ID, Customer_ID, Payment_Mode_ID, CreditCard_Number, Expiry_Date, Name_On_Card, CVV_Number, Payment_Date)**

- This table stores the information of critical credit card details of the customers like credit card number, expiry date, name on card, CVV and Payment_Date. Also, two foreign keys Customer_ID and Payment_Mode_ID are kept in this table to get all relevant information required from Customer_ID. As this is one to one relationship with the customer, One customer can pay by using only one credit card information.

6. **Returned_Item(Return_ID, Return_Date, Return_Description)**

- This table keeps track of the items returned by customers. This stores attributes like Return_Date and Return_Description. Return_Description is a kind of note where item name, quantity and other information can be stored in one attribute only.

7. **Inventory (Item_ID, Name, Color, Size, Description, Price, Available_count)**

- This table stores all the items and every description of those items which can be ordered by the customers. This table can be used to keep track of the stocks sold and reloaded as required. The attributes used in this table are Item name, color, size, description, price and available count of that product.

8. **Order_Mode (Order_Mode_ID, Order_Mode)**

- This table stores the order mode of the order; new entry can be added to this database easily and can be referred in the order table via Order_Mode_ID. The various attributes used in order mode are (Phone, Fax and Mail).

9. **Audit_Trail (Audit_Trail_ID, Item_ID, Trail_DateTime, AuditTransaction_Reason, Trail_Description)**

- This table stores information about every status change on each line item. Accessing this table we can track the whole history of customer's order and the progress for delivering the particular item. Also in this table, foreign key of Item_ID is used to get the item information.

10. **Shipment (Shipment_ID, Shipment_Date)**

- This table stores information about brief shipment details like shipping date. Shipment details only stores the information that is received by the company and automatically updated in the inventory system.

11. **Shipment_Details (Shipment_Detail_ID, *Shipment_ID*, *Item_ID*, *Quantity_Count*)**

- This table stores information of in-depth shipment details of the order. This stores the item information and quantity received in the shipment. For that two-foreign key *Item_ID* and *Shipment_ID* are used to retrieve the information of Item from Item table and *Shipment_ID* from shipment table.

12. **Return_Details (Return_Details_ID, *Return_ID*, *Item_ID*, *Return_Count*, *Return_Description*)**

- This table holds information of the items returned if the customer is not satisfied with the product. This consist of two foreign keys *return_ID* and *Item_ID* that helps to retrieve the information of item.

13. **Payment_Mode(Payment_Mode_ID, *Payment_Mode*);**

- This table stores the payment mode of the order; new entry can be added to this database easily and can be referred in the order table via *Payment_ID*. Various payment mode are by credit card, postal orders and by check.

14. **Other Payment _Info (Payment_ID, *Payment_Mode_ID*, *Customer_ID*, *Cheque_Number*, *PostalOrder_Detail*, *Payment_Date*)**

- This table holds information about other payment modes used. This will mainly store the information of details if the customer uses the payment method via postal orders or checks.

15. **Order_Status (Status_ID, *Status*)**

- This table display the current information about order status. There are four status used i.e Completed, Incomplete, New and Pending.

16. **Delivery_Mode (Delivery_Mode_ID, *Delivery_Mode*)**

- This table stores the delivery type of the order; new entry can be added to this database easily and can be referred in the order table via *Delivery_ID*. The options for delivery mode will be by Post with a basic fee, Express door-to-door delivery and for Heavy orders delivered by SATS.

17. **Canceled_Order(Canceled_ID, *Order_ID*, *Canceled_Date*, *Canceled_Reason*)**

- This table stores the cancelled order information. If a customer wishes to cancel the order, he/she can do so if the product is not shipped. The attributes used in this table are *canceled_date* and *canceled_reason*.

18. **Refund (Refund_ID, *Canceled_ID*, *Order_ID*, *Refund_Amount*, *Refund_Description*)**

- This table stores the refund information. Once the product is returned or order is canceled the company starts to process the refund. This table stores two foreign key canceled_id and order_id. The attributes stored in this table are Refund amount and refund description.

19. **Order_Details (Order_Details_ID, *Order_ID*, *Item_ID*, *Item_Price*, *Personalization*, *Quantity-Ordered*, *Priority*)**

- This table stores details of each order. If a customer requires a specific personalization in the order, that data is stored in this table. This table consist of attributes like Item Price, Personalization information, quantity ordered and priority. There are two foreign key used in this table i.e Order_Id and Item_Id.

5. Final ERD Illustration

The below is the Entity Relationship model diagram proposed for Target Enterprise Pty. Ltd. Database Design. It contains 19 tables after normalization(3NF). The tables are named:

1. Customers, 2. Orders, 3. Inventory, 4. Audit_Trail, 5. Canceled_Order, 6. CreditCard_Info, 7. Delivery_Mode 8. Order_Details 9. Order_Mode 10. Order_Status 11. OtherPayments_Info 12. Payment_Mode 13. Refund 14. Return_Details 15. Returned_Item 16. Shipment 17. Shipment_Details 18. Shipping 19. Shipping_Details.
2. Each table is uniquely identified by a Primary Key. Example, for table Customers, Primary Key is Customer_ID as shown in the image with notation PK. Similarly, in other tables notations for Primary Key is PK.
3. Tables are interconnected to each other using Foreign Key indicated as FK, which are generally the Primary Key of the connected tables.
4. The relation defined between two tables are done using Crow's-Foot notation where “Two vertical line- ||” represent “One and only one” and one similar to “crow foot with a single vertical line” indicates “One or Many”.
5. For example, in above image: A customer “||” can order for one or more items.
6. Similarly, we have shown relation among all the tables using cardinalities. Some of them are listed below.

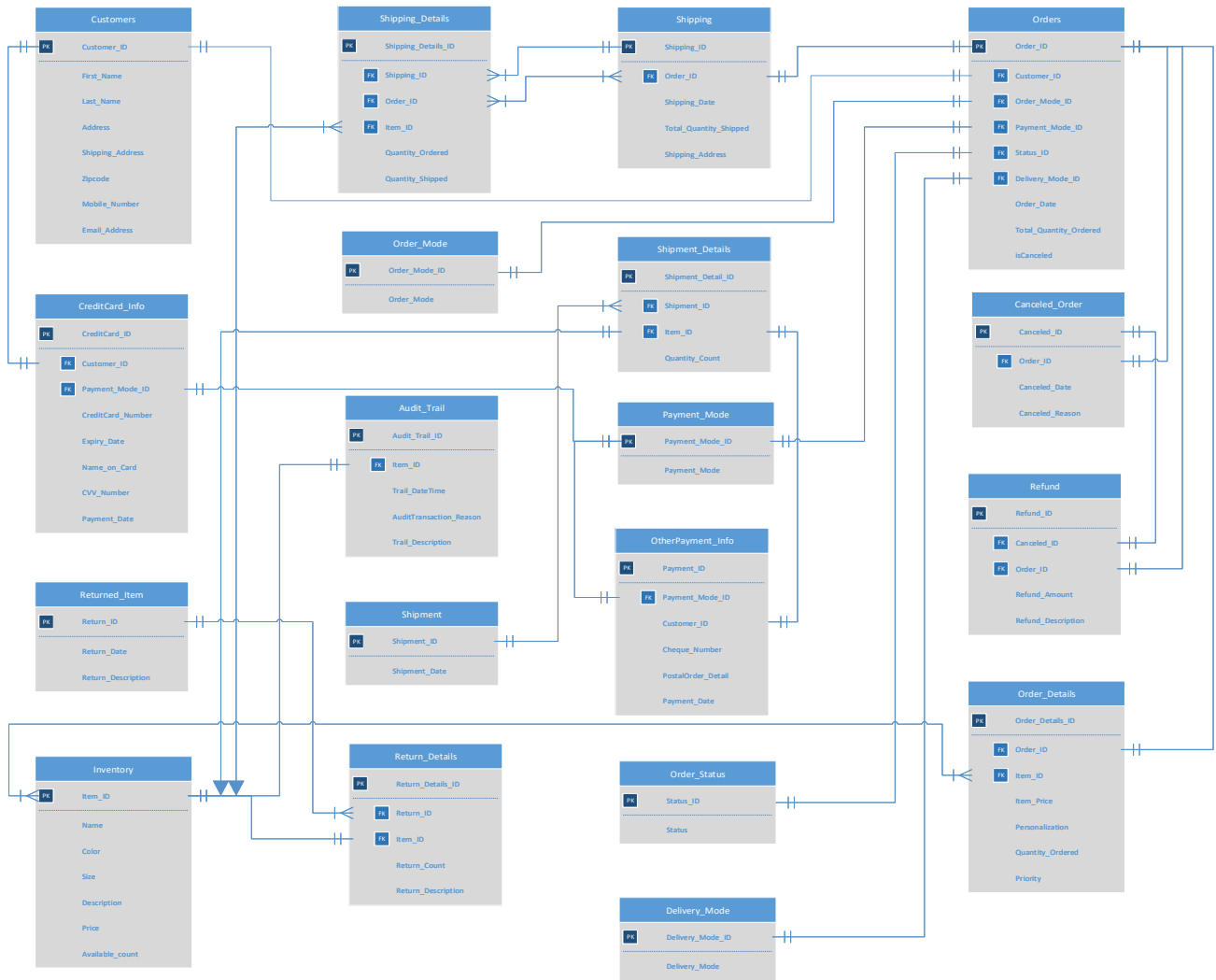


Figure 5.1 Final ERD Diagram

Cardinalities:

1. One Customer can order for one or more items in a single or multiple orders from inventory.
2. One Customer can have only one order mode, payment mode and delivery mode.
3. One Order can have one or more shipping details.
4. One customer can have one and only one credit card details.
5. Similarly other cardinalities are shown in the above ER Diagram.

6. Description of Changes

After successful implementation and completion of tasks in **Part 1** , the team started working on the tasks enlisted in **Part 2** that encountered many changes. All the description of changes are listed below in two parts, first part will consist of improvements required in our proposed system and second part is new functionalities added in our system according to the requirements.

We have updated the Customer & Order tables to improve the functionality of the system.

1. **Customer:** We have added the Shipping address attribute which will enable the system to store the address of the customer. This will help the organisation to maintain updated information of customers as well as the recent address.
2. **Order:** To get a more detailed information of customers like payment, delivery and order details we have added the attributes which will store information like mode of payment, mode of delivery, date of order and finally an attribute which will count the total number of items ordered. Also we created a new table that will store the detailed order information like Quantity Ordered and Items Description for each items ordered. The Order Status will be updated at the end of each day on Click of a button. Thus making it automated as compared to project checkpoint 1, where the administrator was needed to update every order status manually. On clicking the button, SQL query will update the status of every order which is described with an SQL query in the below List of queries section of this document.

Also, we have added many new functionalities in the system and the description is given below according the various parameters used in the system:

1. **Shipping** - As proposed in part 2 it was necessary to add the functionality that will store the overall entry of the order shipped, so we added a brief table To eliminate the redundancy we normalized the tables, thus creating more tables that consist of detailed description provided in Shipping Details table because, one shipping order can have multiple items to be delivered.
2. **Payment Information-** As per the new requirement, the system wanted to store the information of credit card used by customer to purchase the products. To do so we added the functionality that will store information like Credit card number, date of expiry, name on card and CVV. For now, one customer can only store one credit card information.

If the customer has a different payment option other than credit card, this information will store in different table and that will include postal orders and check.

3. **Item Return, Refund and Order Cancellation-** As customers tend to return their item if they are not satisfied with the product. This functionality stores the value of items returned and after that it will increase the count in the inventory and will be considered as old item to process it again or discard it if required.

After returning the item, the refund will be immediately processed and for this functionality we created a table of refund that will connect the functionality of item return and refund, so that there is no issue in tracking down the return and refund process.

For order cancellation, customer can cancel their order any time before the shipping is executed. We have provided three options to the customer i.e cancelling the entire order, cancelling the individual item and order cancellation if the inventory does not have that specific item.

4. **Shipment** - Due to new proposed system, we added a new functionality that will store the information of the shipment that is received by the company to reload the items in the inventory. To avoid redundancies and anomalies we normalized the table, thus creating multiple tables that will store brief information of shipment in one table and detailed information will be stored in another table.

5. **Audit Trail** - As per the requirement, we created a functionality that keeps track of inventory and change in inventory at any point for the reasons mentioned. Whenever there is any change in the inventory an entry will be posted to the database table with the help of triggers with the respective reason. So, any issue can be easily tracked and solved whenever required.

Other changes: To automate the process of Target Enterprise using computerized management system we enhanced the system by providing functionalities address in the business problems section. We were successfully able to implement the business needs and requirements posted in the Phase 1 and Phase 2 of the project.

- To automate the initial process of order feeding into the system, we will provide functionality which will scan the order form and populate the entries in the add order form page, which then be updated as necessary and submitted
- To help the dispatch department and optimizing the printing and packing process, a SQL query is written which will display the order which is to be processed next in line.

- To solve the problem of insufficient stock, a function is created to calculate the count of each line item ordered which will check if has sufficient stock in the inventory.
- To help the Marketing Department in finding duplicate mailing entries, a necessary query solution is created.
- To incorporate the feature where the system can print itemized lists of payments, which have to be sent to the Accounting Department, we have created two stored procedures which will display the order and necessary payment details for both type of payment methods, which can be useful for the accounting department at the end of each working day.

7. List of Queries, Stored Procedures and Triggers

This section will display multiple SQL queries including database creation, table creation and inserting values into tables. Also, it will show the database reach for the computerized order management system for Target Enterprise Pty. Ltd.

The SQL queries for creating database tables and inserting values is provided in TargetSqlScript.txt file.

Sample of creating a table:

Customers information table:

```
CREATE TABLE [dbo].[Customers](
    [Customer_ID] [int] IDENTITY(1,1) NOT NULL,
    [First_Name] [nvarchar](50) NULL,
    [Last_Name] [nvarchar](50) NULL,
    [Address] [nvarchar](50) NULL,
    [Shipping_Address] [nvarchar](50) NULL,
    [ZipCode] [nvarchar](50) NULL,
    [Mobile_Number] [nvarchar](50) NULL,
    [Email_Address] [nvarchar](50) NULL,
    CONSTRAINT [PK_Customer] PRIMARY KEY CLUSTERED
(
    [Customer_ID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
    ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
```

) ON [PRIMARY]

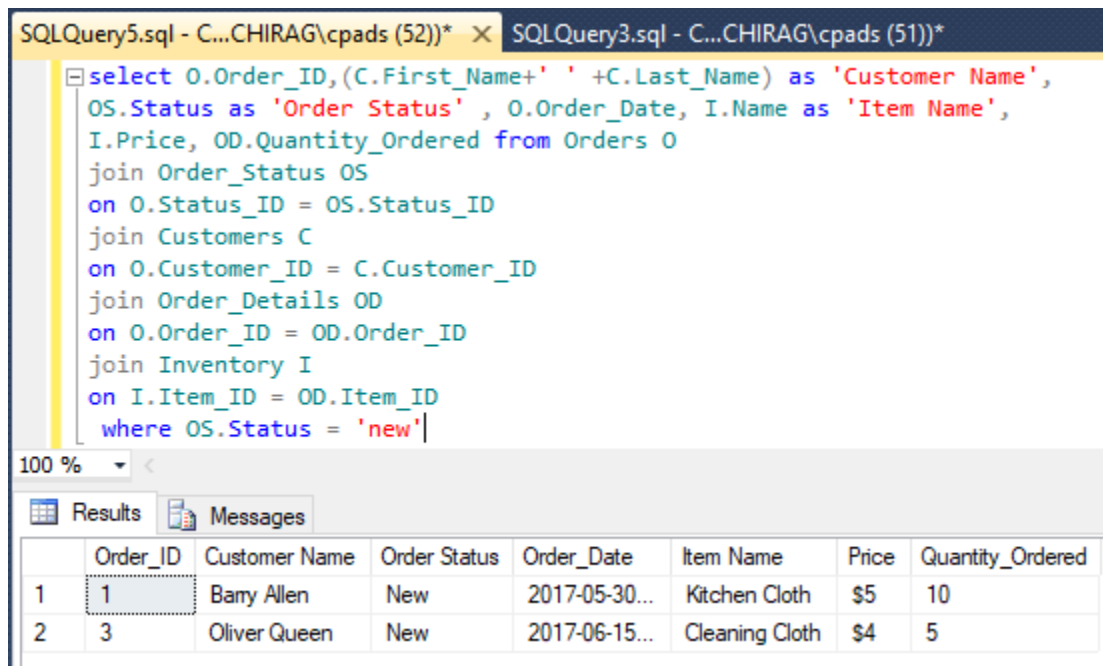
Insert values into Customers Relation:

```
INSERT into Customers VALUES ('John', 'Will', '2408 Nutwood', '2408 Nutwood', '92831', '6576576576',  
'john@j.com')  
( 'Adam', 'Ros', '2400 Nutwood', '2400 Nutwood', '92831', '6576586596', 'adam@a.com')  
, ('Barry', 'Alle', '24 Central City', '24 Central City', '92832', '657658655', 'berry@flash.com')  
, ('Oliver', 'Quee', '10 Lian Yu', '10 Lian Yu', '98233', '6576576569', 'oliver@arrow.com')  
, ('Felicity', 'Smoak', '29 Starling City', '29 Starling City', '92852', '6546586576', 'felicity@arrow.com')  
, ('Jim', 'Gordo', '27 Gotham Mad City', '27 Gotham Mad City', '92854', '6577586576', 'jim@gotham.com')
```

Similarly, we can create tables mentioned in the ERD model and insert values into each table mentioned in the TargetSqlScript.txt.

7.1 Finding Order details for orders which are new.

Below is the screenshot displaying the necessary details of Customers as well as the order placed by them and their order status is new. Similarly, we can find order details for ‘Completed’, ‘Incomplete’ and ‘Pending’. This result can also be used by dispatch department clerk to get upcoming “New” orders which are needed to continue picking and packing process.



The screenshot shows a SQL Server Enterprise Manager window with two tabs: 'SQLQuery5.sql - C:\CHIRAG\cpads (52)*' and 'SQLQuery3.sql - C:\CHIRAG\cpads (51)*'. The active tab displays a SQL query that joins the Orders, Order_Status, Customers, Order_Details, and Inventory tables to find new orders. Below the query editor, the 'Results' pane shows a table with 8 columns: Order_ID, Customer Name, Order Status, Order_Date, Item Name, Price, and Quantity_Ordered. The results table contains two rows of data.

```
select O.Order_ID, (C.First_Name+ ' ' +C.Last_Name) as 'Customer Name',  
OS.Status as 'Order Status' , O.Order_Date, I.Name as 'Item Name',  
I.Price, OD.Quantity_Ordered from Orders O  
join Order_Status OS  
on O.Status_ID = OS.Status_ID  
join Customers C  
on O.Customer_ID = C.Customer_ID  
join Order_Details OD  
on O.Order_ID = OD.Order_ID  
join Inventory I  
on I.Item_ID = OD.Item_ID  
where OS.Status = 'new'
```

	Order_ID	Customer Name	Order Status	Order_Date	Item Name	Price	Quantity_Ordered
1	1	Barry Allen	New	2017-05-30...	Kitchen Cloth	\$5	10
2	3	Oliver Queen	New	2017-06-15...	Cleaning Cloth	\$4	5

Image 7.1 Finding Order details

7.2 Itemized list of payment which can be fetched at the end of each day for the accounting department.

The below stored procedure “spx_GetCreditCardDailyInfo” will be executed on the click of the “Fetch” button for getting info of Customers paying through Credit Card on “Customer Info and Audit Trail” Page of the web application.

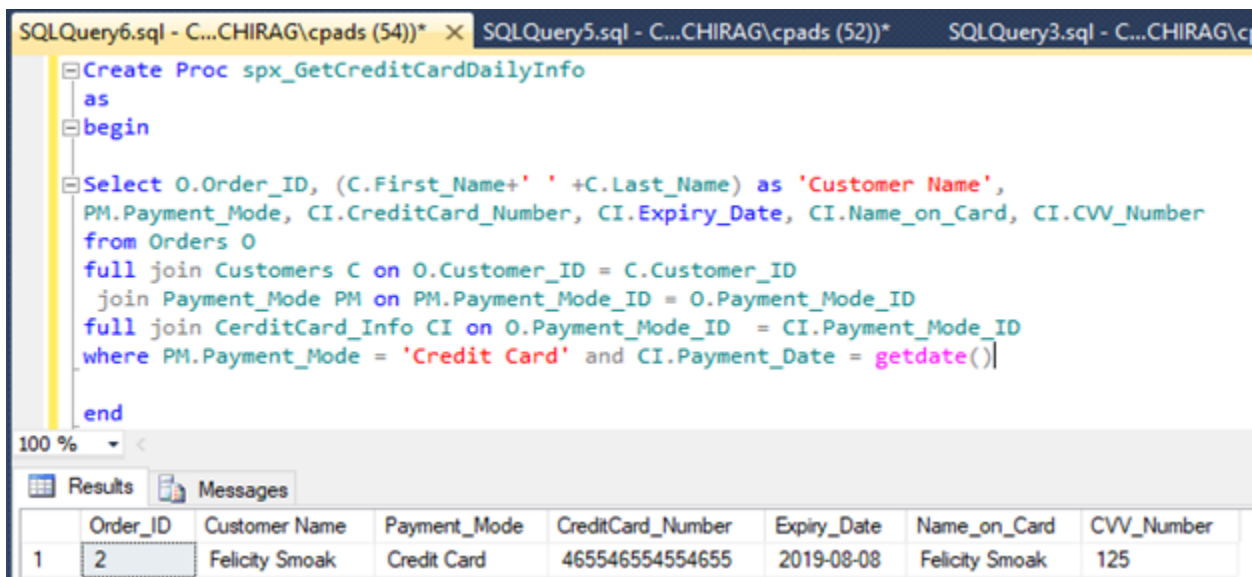


Image 7.2.1 Itemized list of payment for credit card

The below stored procedure “spx_PostalOrder_and_CheckDailyInfo” will be executed on the click of the “Fetch” button for Postal orders and Checks orders on “Customer Info and Audit Trail” Page of the web application.

SQLQuery8.sql - C...\CHIRAG\cpads (59))* X SQLQuery7.sql - C...\CHIRAG\cpads (56))* SQLQuery6.sql - C...\CHIRAG\cpads (55))*

```

USE [TargetEnterprise]
GO
/***** Object: StoredProcedure [dbo].[spx_PostalOrder_and_CheckDailyInfo]
Script Date: 6/26/2017 10:15:13 AM *****/
ALTER Proc [dbo].[spx_PostalOrder_and_CheckDailyInfo]
as
begin
|
Select O.Order_ID, (C.First_Name+' ' +C.Last_Name) as 'Customer Name', PM.Payment_Mode,
OP.PostalOrder_Detail, OP.Cheque_Number
from Orders O
full join Customers C on O.Customer_ID = C.Customer_ID
join Payment_Mode PM on PM.Payment_Mode_ID = O.Payment_Mode_ID
full join OtherPayment_Info OP on OP.Payment_Mode_ID = O.Payment_Mode_ID
where PM.Payment_Mode_ID in (2,3) -- or PM.Payment_Mode in ('Postal Order','Check')
and OP.Payment_Date = getdate()
end

```

100 %

Results Messages

	Order_ID	Customer Name	Payment_Mode	PostalOrder_Detail	Cheque_Number
1	1	Barry Allen	Check	N/A	212121
2	3	Oliver Queen	Postal Order	PN: 254654	N/A

Image 7.2.2 Itemized list of payment for postal Order/cheque Number

7.3 Finding details of each Shipment received.

The below result will be achieved when the admin will insert 'From date' and 'To date' values in the Shipment page of web application and click the search button. Basically, it will display all the shipments received between the dates specified by the admin using the system.

The screenshot displays a SQL Server Enterprise Manager window with three tabs: SQLQuery10.sql, SQLQuery9.sql, and SQLQuery8.sql. The active tab, SQLQuery10.sql, contains the following T-SQL script:

```
USE [TargetEnterprise]
GO
/***** Object: StoredProcedure [dbo].[spx_FindShipmentDetail_ByDate]    Script Da
ALTER proc [dbo].[spx_FindShipmentDetail_ByDate]
@FromDate as datetime, --input parameter from shipment page of web app
@ToDate as datetime    --input parameter from shipment page of web app
as
begin
Select S.Shipment_ID as 'Shipment No.', S.Shipment_Date as 'Shipment Received on',
SD.Item_ID as 'Item Code Received', SD.Quantity_Count from Shipment S
join Shipment_Details SD on S.Shipment_ID = SD.Shipment_ID
where S.Shipment_Date between @FromDate and @ToDate
end;
exec spx_FindShipmentDetail_ByDate '05-06-2017','07-07-2017';
```

Below the script, the 'Results' tab shows the output of the stored procedure execution. The results are displayed in a table with the following columns: Shipment No., Shipment Received on, Item Code Received, and Quantity_Count.

	Shipment No.	Shipment Received on	Item Code Received	Quantity_Count
1	1	2017-06-23 16:44:12.187	2	100
2	1	2017-06-23 16:44:12.187	3	120
3	1	2017-06-23 16:44:12.187	4	150
4	2	2017-06-25 16:45:36.233	3	100
5	2	2017-06-25 16:45:36.233	1	120

Image 7.3 Finding Shipment details

7.4 Trigger to insert data into “Audit_Trail” Table.

The below figure illustrates that an entry will be inserted “Audit_Trail” table whenever there is an Inventory reload when shipment is received, as needed in the system.

SQLQuery12.sql -...CHIRAG\cpads (65))* X SQLQuery11.sql -...CHIRAG\cpads (64))* SQLQuery10.sql -...CHIRAG\cpads (6)

```

USE [TargetEntprise]
GO
/***** Object: Trigger [dbo].[TRIG_AuditTrail]    Script Date: 6/26/2017 11:31:02 AM *****/
ALTER TRIGGER [dbo].[TRIG_AuditTrail]
ON [dbo].[Inventory]
AFTER INSERT
AS
BEGIN
declare @ItemID int;
set @ItemID = (select top 1 I.Item_ID from Inventory I
join Shipment_Details SD on SD.Item_ID = I.Item_ID
join Shipment S on S.Shipment_ID = SD.Shipment_ID
order by S.Shipment_Date asc)

Insert into Audit_Trail values(getdate(),'Inventory reload', 'Shipment received', @ItemID);

END
select * from Audit_Trail where AuditTransaction_Reason = 'Inventory reload'

```

100 %

Results Messages

	Audit_Trail_ID	Trail_DateTime	AuditTransaction_Reason	Trail_Description	Item_ID
1	3	2017-06-23 16:44:12.187	Inventory reload	Shipment received	2
2	4	2017-06-23 16:44:12.187	Inventory reload	Shipment received	3
3	5	2017-06-23 16:44:12.187	Inventory reload	Shipment received	4
4	6	2017-06-25 16:45:36.233	Inventory reload	Shipment received	3
5	7	2017-06-25 16:45:36.233	Inventory reload	Shipment received	1
6	10	2017-06-26 11:29:06.333	Inventory reload	Shipment received	2

Image 7.4.1 Trigger to insert data into “Audit_Trail” Table

Similarly, we can create triggers to insert entry into “Audit_Trail” table whenever: 1. An item is returned as “Item Returned”, 2. Stock are adjusted as “Stock Adjustment”, 3. An item is shipped as “Delivered to Customer”.

We can see the example in the below screenshot of Audit_Trail table displaying the above explained entries.

SQLQuery12.sql -...CHIRAG\cpads (65))* SQLQuery11.sql -...CHIRAG\cpads (64))* SQLQuery10.sql -...

```
select * from Audit_Trail
```

100 %

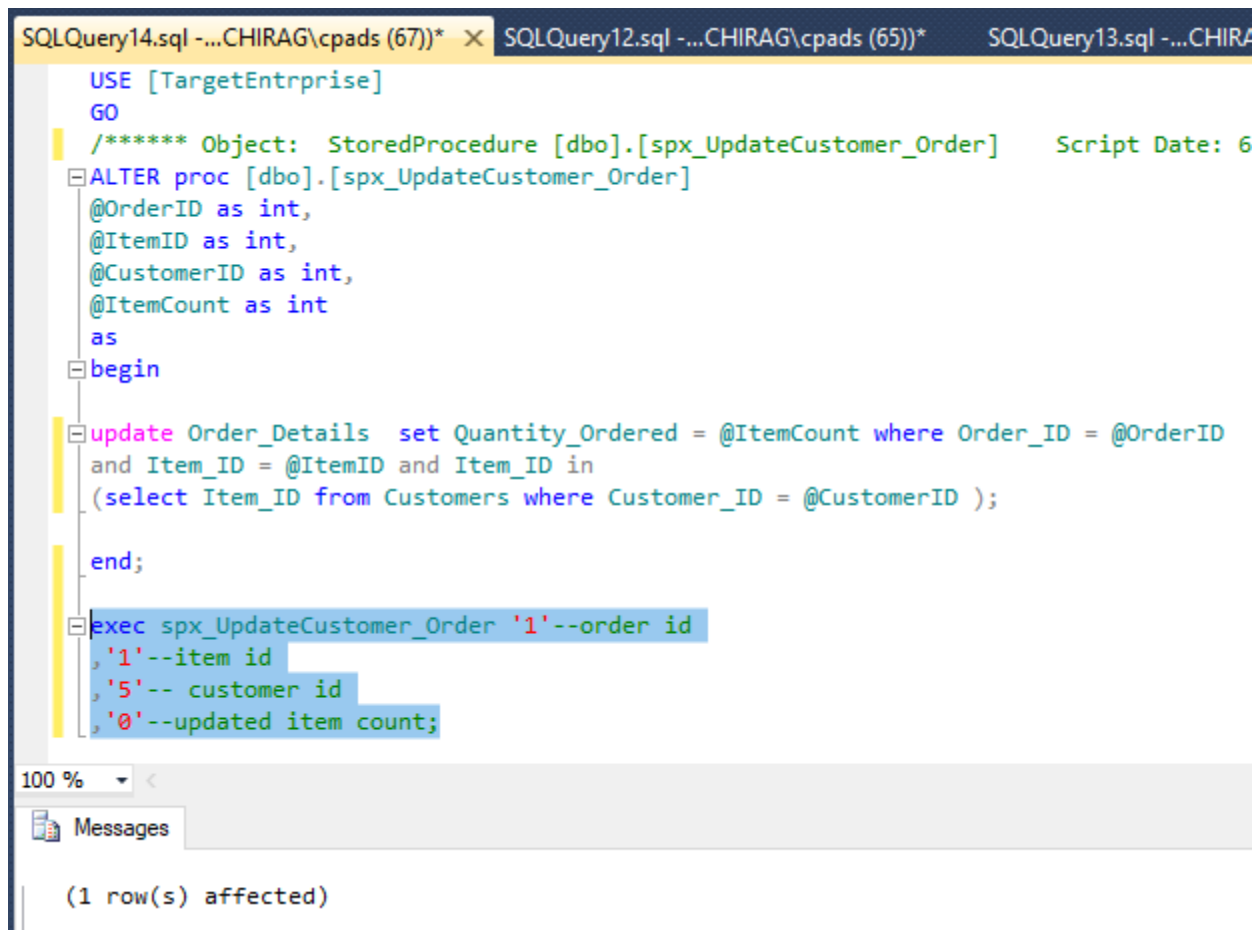
Results Messages

	Audit_Trail_ID	Trail_DateTime	AuditTransaction_Reason	Trail_Description	Item_ID
1	1	2017-06-05 00:00:00.000	Delivered to customer	Delivered to Cust Id 7	2
2	2	2017-06-05 00:00:00.000	Delivered to customer	Delivered to Cust Id 7	3
3	3	2017-06-23 16:44:12.187	Inventory reload	Shipment received	2
4	4	2017-06-23 16:44:12.187	Inventory reload	Shipment received	3
5	5	2017-06-23 16:44:12.187	Inventory reload	Shipment received	4
6	6	2017-06-25 16:45:36.233	Inventory reload	Shipment received	3
7	7	2017-06-25 16:45:36.233	Inventory reload	Shipment received	1
8	8	2017-06-20 00:00:00.000	Returned	Returned by cust id 7	2
9	10	2017-06-26 11:29:06.333	Inventory reload	Shipment received	2

Image 7.4.2 Audit Trail entries

7.5 Updating order over phone for insufficient stock.

If there is any insufficient stock for an item a customer is called to ask if he/she needs the number of items there in the inventory or if he/she wishes to cancel that item line. At this point of time the below type of stored procedure will be called. On the home page when admin enters “Order ID” and click track, he is displayed the order form of that respective customer where he can edit the entries answered by the customer on phone and click update to update the order, as shown in the image below.

The image shows a screenshot of the SQL Server Enterprise Manager interface. At the top, there are three tabs: 'SQLQuery14.sql - ...CHIRAG\cpads (67))*', 'SQLQuery12.sql - ...CHIRAG\cpads (65))*', and 'SQLQuery13.sql - ...CHIRAG\cpads (66))*'. The active tab is 'SQLQuery14.sql'. The main window displays a SQL script for a stored procedure named 'spx_UpdateCustomer_Order'. The script starts with 'USE [TargetEnterprise]' and 'GO'. It then defines the procedure with parameters: '@OrderID as int,', '@ItemID as int,', '@CustomerID as int,', and '@ItemCount as int'. The procedure body begins with 'begin' and contains an 'update' statement for 'Order_Details' table, setting 'Quantity_Ordered' to '@ItemCount' where 'Order_ID = @OrderID' and 'Item_ID = @ItemID'. A subquery is used to select 'Item_ID' from 'Customers' where 'Customer_ID = @CustomerID'. The procedure ends with 'end;'. Below the script, there is an 'exec' statement calling 'spx_UpdateCustomer_Order' with values '1' for order id, '1' for item id, '5' for customer id, and '0' for updated item count. The bottom status bar shows '(1 row(s) affected)'.

```
USE [TargetEnterprise]
GO
/***** Object: StoredProcedure [dbo].[spx_UpdateCustomer_Order]    Script Date: 6/1/2016 10:10:10 AM *****/
ALTER proc [dbo].[spx_UpdateCustomer_Order]
    @OrderID as int,
    @ItemID as int,
    @CustomerID as int,
    @ItemCount as int
as
begin
    update Order_Details set Quantity_Ordered = @ItemCount where Order_ID = @OrderID
    and Item_ID = @ItemID and Item_ID in
    (select Item_ID from Customers where Customer_ID = @CustomerID );
end;
exec spx_UpdateCustomer_Order '1'--order id
    , '1'--item id
    , '5'-- customer id
    , '0'--updated item count;
```

100 % <

Messages

(1 row(s) affected)

Image 7.5 Updating orders for insufficient stock

7.6 Cancelled orders with refund:

The below image displays the result of three query, justifying the refund of the cancelled order by customer itself before shipping. Every query displays appropriate data attributes.

SQLQuery15.sql - ...CHIRAG\cpads (52))" × SQLQuery3.sql - not connected" SQLQuery14.sql - not connected" SQLQuery12.s

```

select O.Order_ID, (C.First_Name+ ' '+C.Last_Name) as 'Customer Name', C.Mobile_Number, OD.Item_ID,
OD.Quantity_Ordered, O.Order_Date, O.isCanceled, CO.Canceled_Date from Orders O
join Customers C on C.Customer_ID = O.Customer_ID
join Order_Details OD on OD.Order_ID = O.Order_ID
join Canceled_Order CO on CO.Order_ID = O.Order_ID
where isCanceled = 1;

select R.Refund_ID, O.Order_ID, (C.First_Name+ ' '+C.Last_Name) as 'Customer Name', OD.Item_ID,
I.Price as 'Each Price', R.Refund_Amount, R.Refund_Description from Refund R
join Orders O on R.Order_ID = O.Order_ID
join Customers C on C.Customer_ID = O.Customer_ID
join Order_Details OD on OD.Order_ID = O.Order_ID
join Inventory I on I.Item_ID = OD.Item_ID;

select O.Order_ID, CO.Canceled_Date, CO.Canceled_Reason from Canceled_Order CO
join Orders O on O.Order_ID = CO.Order_ID
where O.isCanceled = 1;

```

100 %

Results Messages

Order_ID	Customer Name	Mobile_Number	Item_ID	Quantity_Ordered	Order_Date	isCanceled	Canceled_Date
1	Barry Allen	6576586555	1	10	2017-05-30 ...	1	2017-06-01 ...

Refund_ID	Order_ID	Customer Name	Item_ID	Each Price	Refund_Amount	Refund_Description
1	1	Barry Allen	1	\$5	\$50	Refunded the charged amount.

Order_ID	Canceled_Date	Canceled_Reason
1	2017-06-01 0...	Canceled by customer.

Image 7.6 Refund of cancelled orders

7.7 Finding Customer Mailing list for sending promotion offers with non-repeated entries:

Below is the screenshot showing the necessary information of Customers to whom promotional offer /pamphlet/Magazines can be mailed to. It will result only distinct shipping address thus avoiding multiple entries as needed. It will display only a single row for multiple repetitive entries from the database.

SQLQuery3.sql - C...CHIRAG\cpads (51))* X

```

select distinct C.Shipping_Address,
(C.First_Name+C.Last_Name) as 'Customer Name' ,
C.Mobile_Number, C.Email_Address from Customers C

```

100 % <

Results Messages

	Shipping_Address	Customer Name	Mobile_Number	Email_Address
1	10 Lian Yu	OliverQueen	6576576569	oliver@arrow.com
2	24 Central City	BarryAllen	6576586555	berny@flash.com
3	2400 Nutwood	AdamRos	6576586596	adam@a.com
4	2408 Nutwood	JohnWill	6576576576	john@j.com
5	27 Gotham Mad ...	JimGordon	6577586576	jim@gotham.com
6	29 Starling City	FelicitySmoak	6546586576	felicity@arrow.c...

Image 7.7 Finding distinct mailing list

7.8 Finding bestselling item of the organization:

The below query will display the bestselling item in the organization. It can display top 1,2,3, etc. or even all item sold by the organization with the count of numbers of items sold. Below image select top 2 and all the items as shown in the image(s) below.

SQLQuery17.sql -...CHIRAG\cpads (56))* × SQLQuery16.sql -...CHIRAG\cpads (54))* × SQLQuery15.sql -...CHIRAG

```

begin
create table #temp(
    Item_ID int,
    Best_Item nvarchar(50),
    Quantity_Sold int
)

declare @InvCount int;
set @InvCount =( select Count(*) from Inventory)

declare @i int;
set @i = 0;

while @i < @InvCount +1
begin
insert into #temp (Item_ID,Best_Item,Quantity_Sold)
select I.Item_ID as 'Item_ID' , I.Name as 'Best_Item' , sum(SD.Quantity_Shipped) as
'Quantity_Sold' from Shipping_Details SD
join Inventory I on I.Item_ID = SD.Item_ID
where SD.Item_ID = @i group by I.Item_ID, I.Name;
set @i = @i + 1;
end;

select * from #temp order by Quantity_Sold desc;
drop table #temp;
end

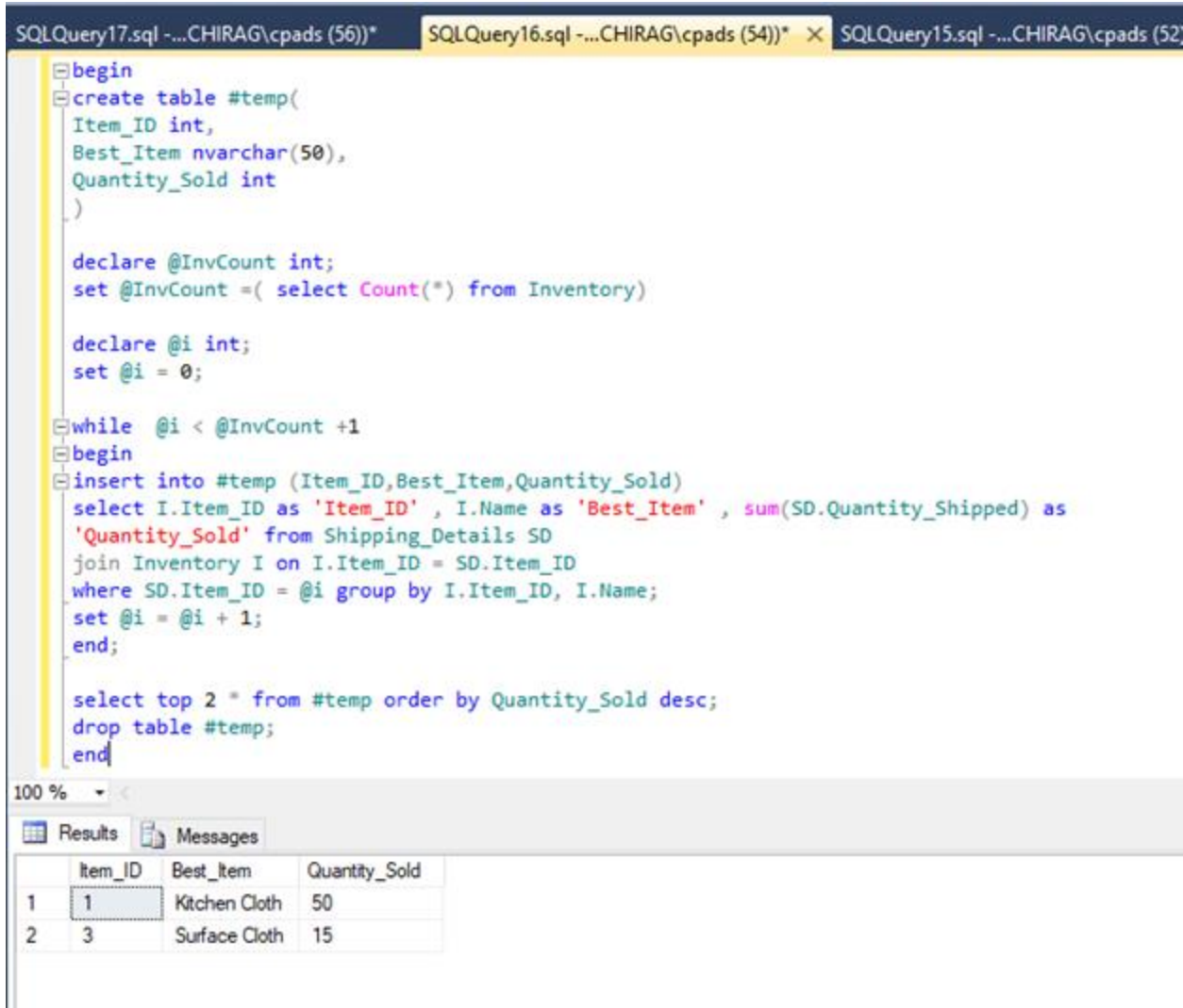
```

100 % <

Results Messages

	Item_ID	Best_Item	Quantity_Sold
1	1	Kitchen Cloth	50
2	3	Surface Cloth	15
3	2	Handkerchief	12
4	4	Cleaning Cloth	5

Image 7.8.1 Finding bestselling item



```

SQLQuery17.sql -...CHIRAG\cpads (56))*  SQLQuery16.sql -...CHIRAG\cpads (54))*  SQLQuery15.sql -...CHIRAG\cpads (52)
begin
create table #temp(
Item_ID int,
Best_Item nvarchar(50),
Quantity_Sold int
)

declare @InvCount int;
set @InvCount =( select Count(*) from Inventory)

declare @i int;
set @i = 0;

while @i < @InvCount +1
begin
insert into #temp (Item_ID,Best_Item,Quantity_Sold)
select I.Item_ID as 'Item_ID' , I.Name as 'Best_Item' , sum(SD.Quantity_Shipped) as
'Quantity_Sold' from Shipping_Details SD
join Inventory I on I.Item_ID = SD.Item_ID
where SD.Item_ID = @i group by I.Item_ID, I.Name;
set @i = @i + 1;
end;

select top 2 * from #temp order by Quantity_Sold desc;
drop table #temp;
end

```

100 %

Results Messages

	Item_ID	Best_Item	Quantity_Sold
1	1	Kitchen Cloth	50
2	3	Surface Cloth	15

Image 7.8.2 Finding bestselling item by quantity sold

7.9 Print upcoming order which needs to be served:

For finding the next order information- “Print” button on the home page will be clicked, which will fire the below query, which will return top 1 entry from the table, but the below image just shows all entries which can be altered by using “Select top 1 *”.

SQLQuery18.sql -...CHIRAG\cpads (57))* X SQLQuery17.sql -...CHIRAG\cpads (56))* SQLQuery16.sql -..

```

select O.Order_ID, (C.First_Name+' '+C.Last_Name) as 'Customer Name',
C.Mobile_Number, OD.Item_ID,
OD.Quantity_Ordered, O.Order_Date, OD.Priority from Orders O
join Customers C on C.Customer_ID = O.Customer_ID
join Order_Details OD on OD.Order_ID = O.Order_ID
where OD.Priority = 'high'

```

100 %

Results Messages

	Order_ID	Customer Name	Mobile_Number	Item_ID	Quantity_Ordered	Order_Date	Priority
1	2	Felicity Smoak	6546586576	2	5	2017-06-01 ...	high
2	2	Felicity Smoak	6546586576	3	5	2017-06-01 ...	high
3	3	Oliver Queen	6576576569	4	5	2017-06-15 ...	high

Image 7.9 Print upcoming order

7.10 Shipping Details:

The below image shows the detail of shipped orders. It displays appropriate data such as Shipping Date, Quantity Ordered, Quantity Shipped etc.

SQLQuery18.sql -...CHIRAG\cpads (57))* X CHIRAG.TargetEntrp...o.Shipping_Details CHIRAG.TargetEnterprise - dbo.Inventory S

```

Select distinct S.Shipping_ID ,(C.First_Name+' '+C.Last_Name) as 'Customer Name',
S.Shipping_Address, S.Shipping_Date ,SD.Item_ID,SD.Quantity_Ordered , SD.Quantity_Shipped
from Shipping_Details SD
join Orders O on O.Order_ID = SD.Order_ID
join Customers C on C.Customer_ID = O.Customer_ID
right join Shipping S on S.Shipping_ID = SD.Shipping_ID

```

100 %

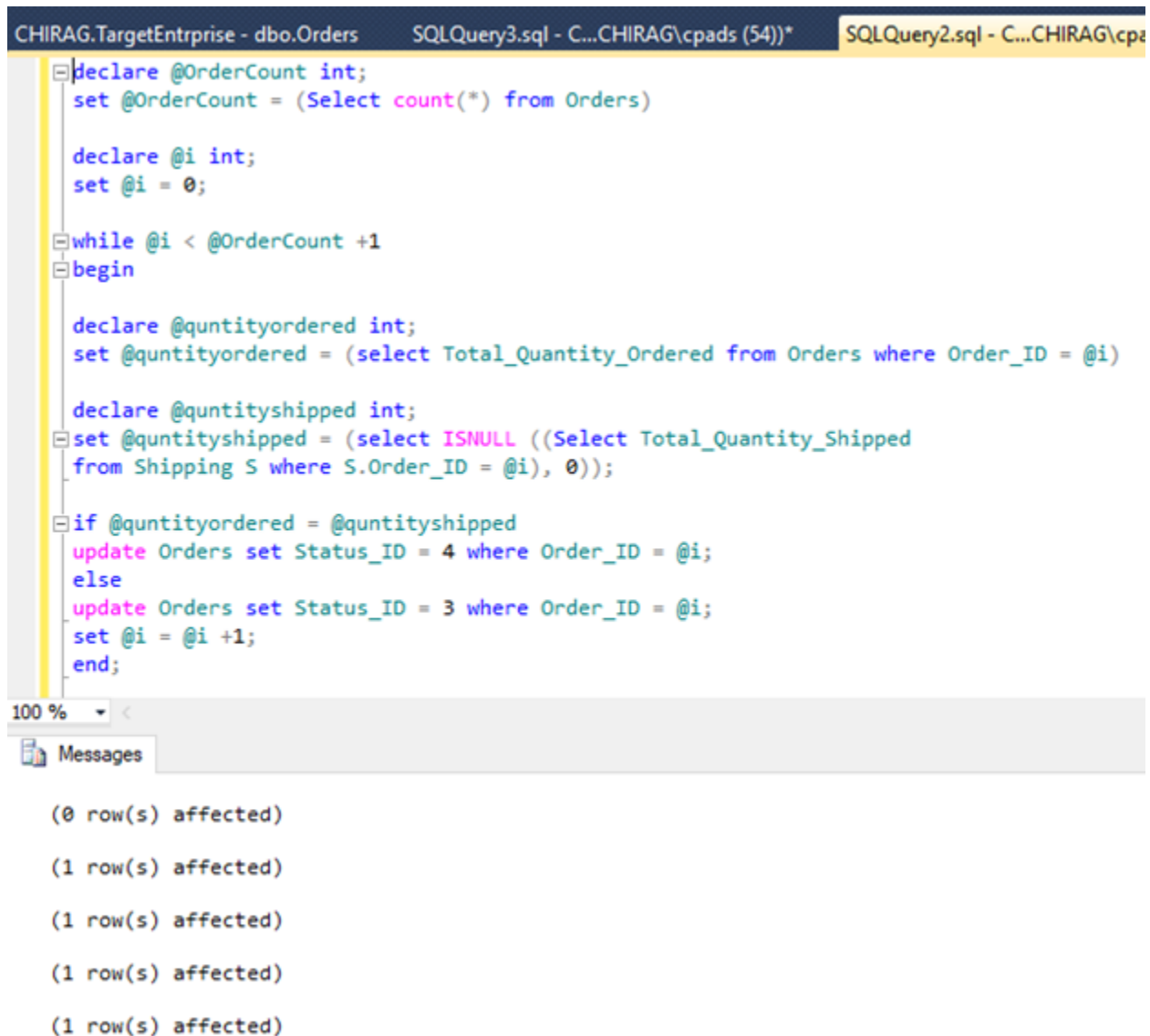
Results Messages

	Shipping_ID	Customer Name	Shipping_Address	Shipping_Date	Item_ID	Quantity_Ordered	Quantity_Shipped
1	1	Felicity Smoak	29 Starling City 92852	2017-06-05 00:00:00.000	2	5	5
2	1	Felicity Smoak	29 Starling City 92852	2017-06-05 00:00:00.000	3	5	5
3	2	Oliver Queen	10 Lian Yu	2017-06-25 00:00:00.000	1	10	10
4	2	Oliver Queen	10 Lian Yu	2017-06-25 00:00:00.000	4	5	5
5	4	John Will	2408 Nutwood	2017-07-01 00:00:00.000	1	15	15
6	4	John Will	2408 Nutwood	2017-07-01 00:00:00.000	2	2	2
7	5	Adam Ros	2400 Nutwood	2017-07-02 00:00:00.000	1	25	25
8	5	Adam Ros	2400 Nutwood	2017-07-02 00:00:00.000	2	5	5
9	5	Adam Ros	2400 Nutwood	2017-07-02 00:00:00.000	3	10	10

Image 7.10 Shipping Details

7.11 Update Order Status Automatically On click of button:

By clicking update button on the web application this query will be fired, which will update all the order status which are delivered by the end of the day. The ordered which are delivered will be set to “Complete” and rest orders will be updated as “Incomplete” thus keeping a differentiation between “New” and “Incomplete”.



The screenshot displays the SQL Server Enterprise Manager interface. The top pane shows a T-SQL query in the 'SQLQuery3.sql' file. The query is as follows:

```
declare @OrderCount int;
set @OrderCount = (Select count(*) from Orders)

declare @i int;
set @i = 0;

while @i < @OrderCount +1
begin

    declare @quantityordered int;
    set @quantityordered = (select Total_Quantity_Ordered from Orders where Order_ID = @i)

    declare @quantityshipped int;
    set @quantityshipped = (select ISNULL ((Select Total_Quantity_Shipped
    from Shipping S where S.Order_ID = @i), 0));

    if @quantityordered = @quantityshipped
    update Orders set Status_ID = 4 where Order_ID = @i;
    else
    update Orders set Status_ID = 3 where Order_ID = @i;
    set @i = @i +1;
end;
```

The bottom pane shows the 'Messages' tab with the following output:

```
(0 row(s) affected)
(1 row(s) affected)
(1 row(s) affected)
(1 row(s) affected)
(1 row(s) affected)
```

Image 7.11 Update order status automatically

8. Discuss Future Scope

- In the future, Target Enterprises (Pty) Ltd can also undertake other business intelligence or data warehouse solutions as the company grows.
- The system can be enhanced to identify which clerk falls under which departments of the organization. Thus, orders executed by clerks can also be tracked easily.
- Customers complaints can be handled easily by knowing the clerk which served them.
- To target new customers and retain existing customer database, various marketing Campaigns can be devised such as sending out email blasts and developing customer surveys.
- Target enterprise can also choose to create a new E-Commerce website to sell products online and deliver the products to other countries by expanding their business.
- The company can also advertise using Google Adwords and many other online campaign for marketing the company online.
- The company can develop a LIVE HELP technical service with a dedicated team of employees who can assist the customers in addressing their queries.
- A section for “Frequently Asked Questions” which will cover frequently asked questions can be included.
- Using the purchase history of customers, the system can use Business Analytics to recommend related products which the customers are most likely to buy in near future.
- The system can be improved by providing a feature which will enable the customers to track the package in real time.

9. Web App Prototype

9.1. Home Page:

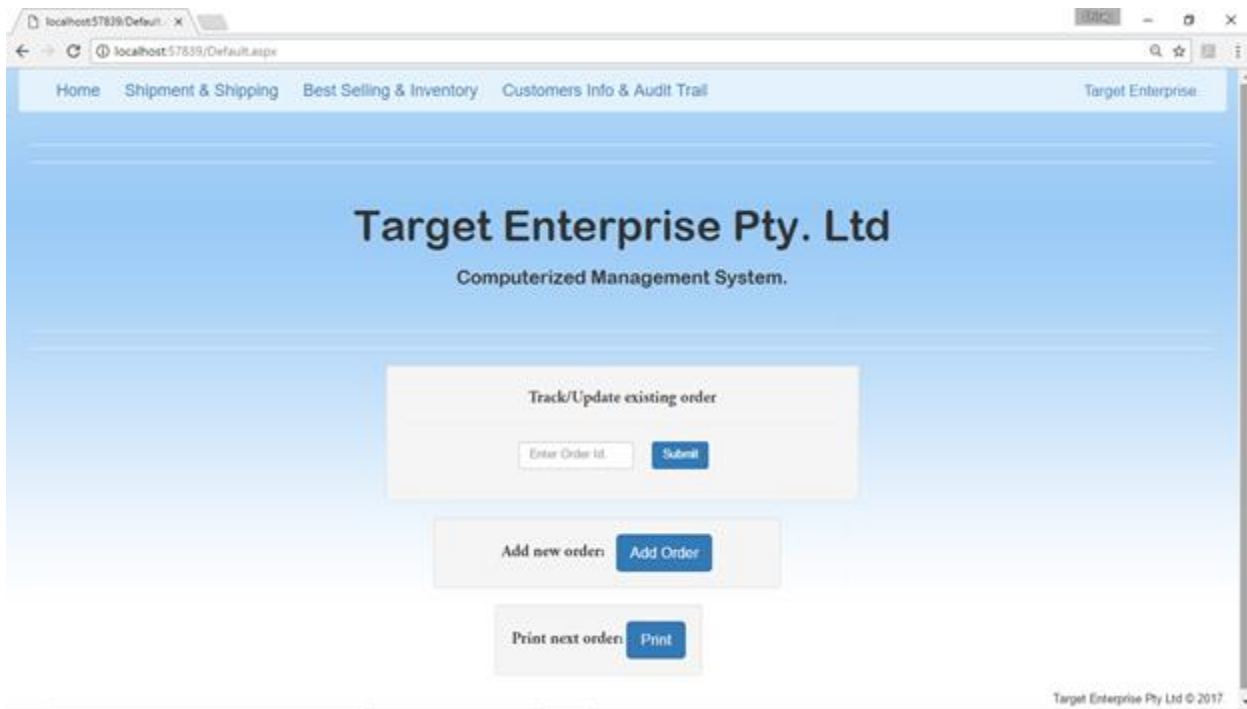


Image 9.1 Home page

The above image is the home page of Computerized management system for Target Enterprise Pty. Ltd. The navigation bar consists of 4 different tabs. 1. Home, 2. Shipment & Shipping, 3. Best Selling & Inventory, 4. Customer Info & Audit Trail, which will be explained below.

The Track/Update existing order on the home page above- Ask for Order Id which will then provide the information of User's order, which will be in editable format. The order can be updated or cancelled as per customer's request. It can be helpful while asking a customer about insufficient stock of the item ordered, if customer wishes to cancel particular item line or the whole order, the admin can easily edit the order using this tab.

The Add order will redirect the admin to the new page which is shown below.

The Print next order will print the next order by firing the query shown in 7.9 and its query in Image 7.12.

9.2. Add Order Page:

Order Form.

Scan Image

Contact Info

Enter First name Enter Last name Home Contact Shipping Contact

Residence Address

Street Name City State Zip Code

Shipping Address ☐ Same as Residence Address

Street Name City State Zip Code

Order Info

Item Code Item Color Item Size Item Description

Personalized Text Quantity ordered Price Each Total Price

Add Item Line

Submit/Update Form Cancel

Target Enterprise Pty Ltd © 2017

Image 9.2 Add order page

The above image displays the new order form. The admin can insert the values manually or by scanning the order form which will populate all the entries in the order form. Once the entries are populated admin can submit the form, thus creating order entry in the database. If there are multiple lines admin can click add line to add another order line if entering the order manually.

9.3. Shipping and Shipment Info:

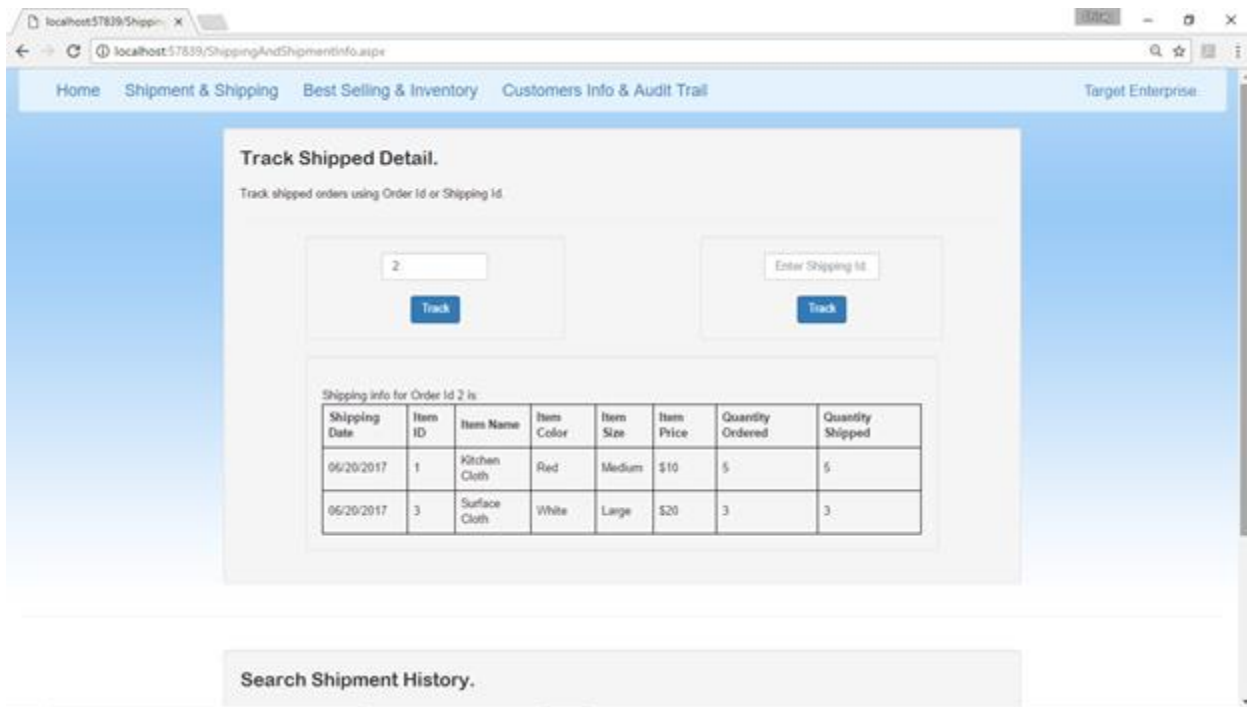


Image 9.3.1 Shipping info

The above image shows how the admin can fetch the shipped order data. He/she can do this by entering either Shipping Id or Order Id as shown in the above Image. After submitting the shipping or order Id the web app will populate the appropriate data as shown in the image above. The query example can be seen the Image 7.13 above.

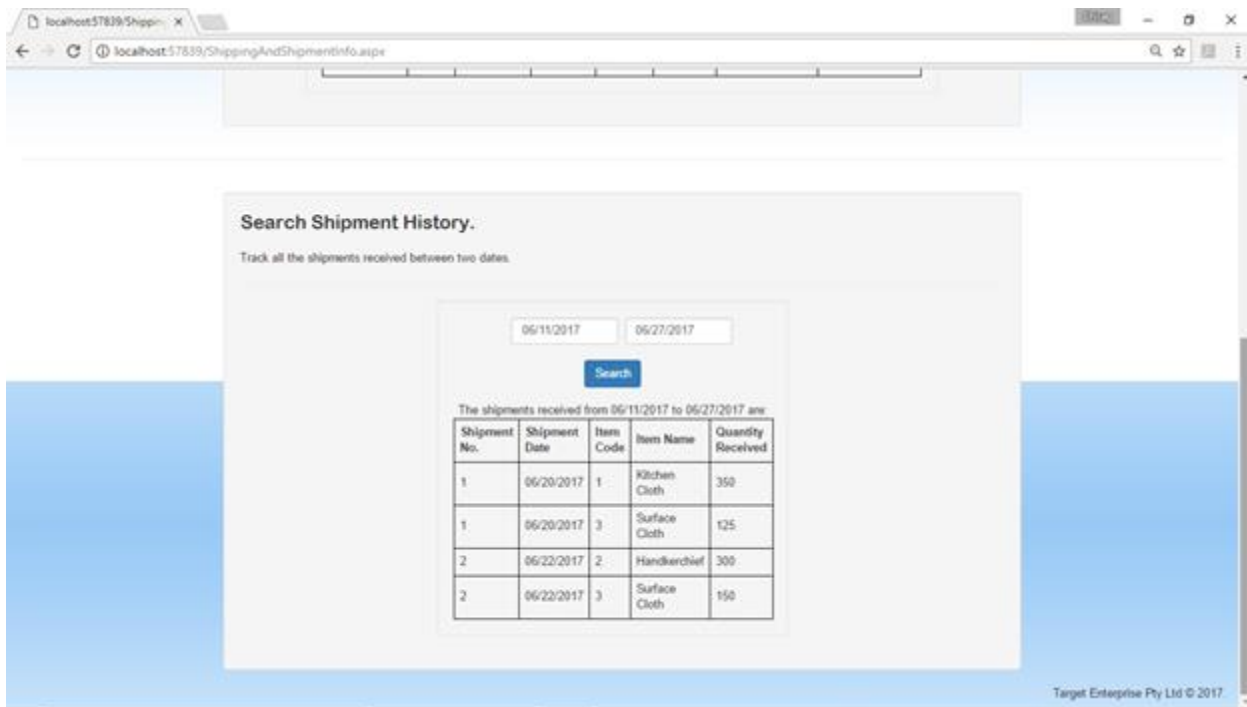


Image 9.3.2 Shipment info

As shown in the image above the Admin can keep track of all the shipments received by entering two dates which will display all the records of shipments received between those two specified dates. The above image also shows an example of this.

9.4. Best Selling Item and Inventory Info:

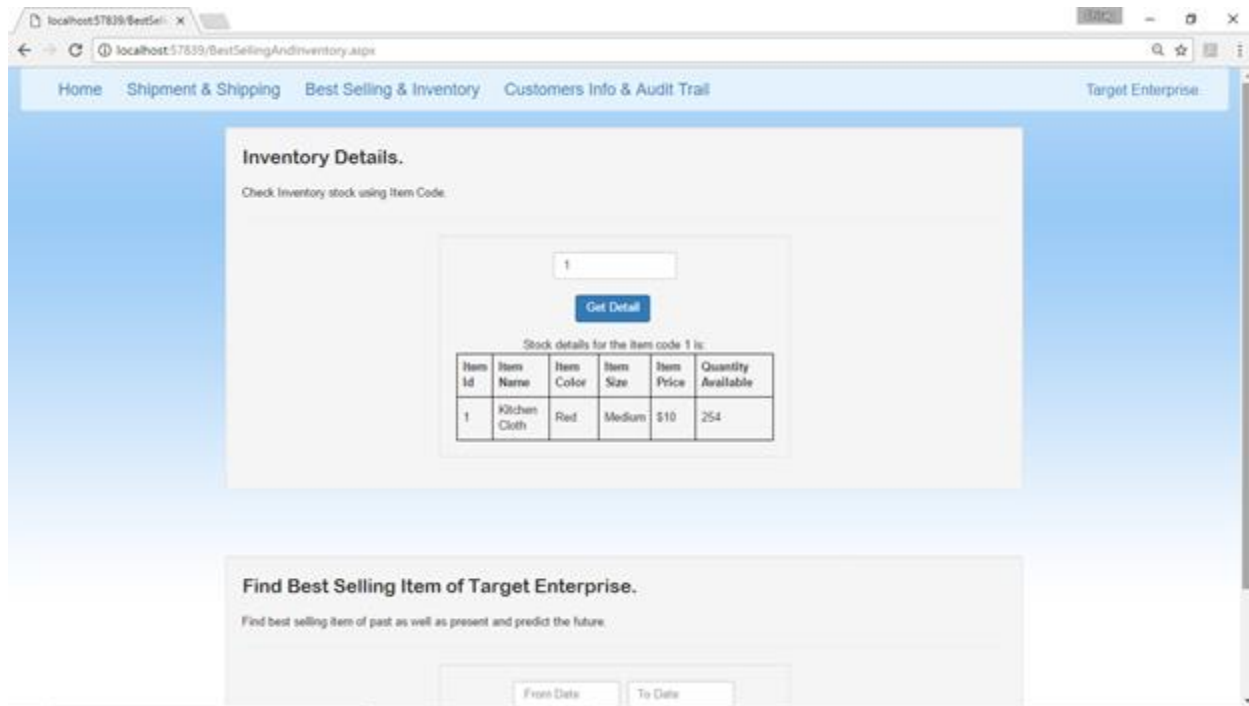


Image 9.4.1 Bestselling item info

The above image shows Inventory part of the web application, where how many items are there in the inventory can be figured out by the admin by entering the Item Id. As shown in the above image, It displays the stock available for Item Id 1.

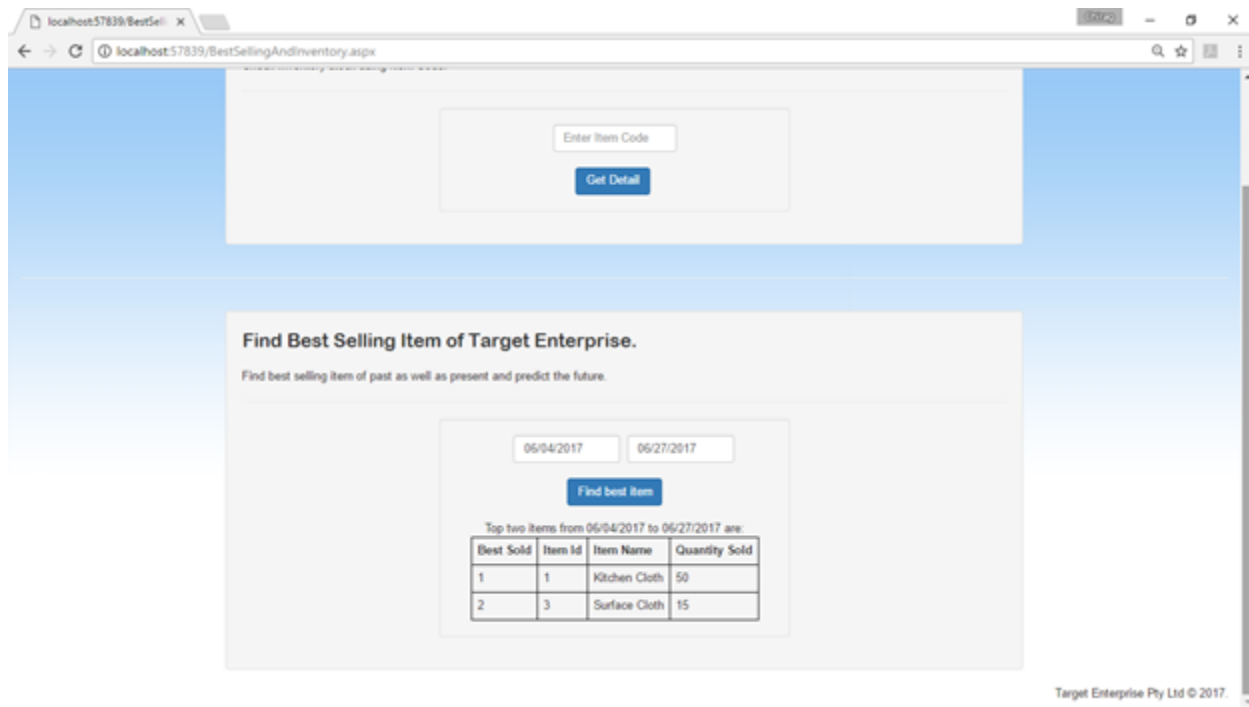


Image 9.4.2 Inventory info

The above image shows the best item sold by the organization between the specified date. It will display top 2 item as shown in the image 7.11 above. This can help the Target Enterprise with printing promotional/advertisement pamphlets.

9.5. Customer Info and Audit Trail:

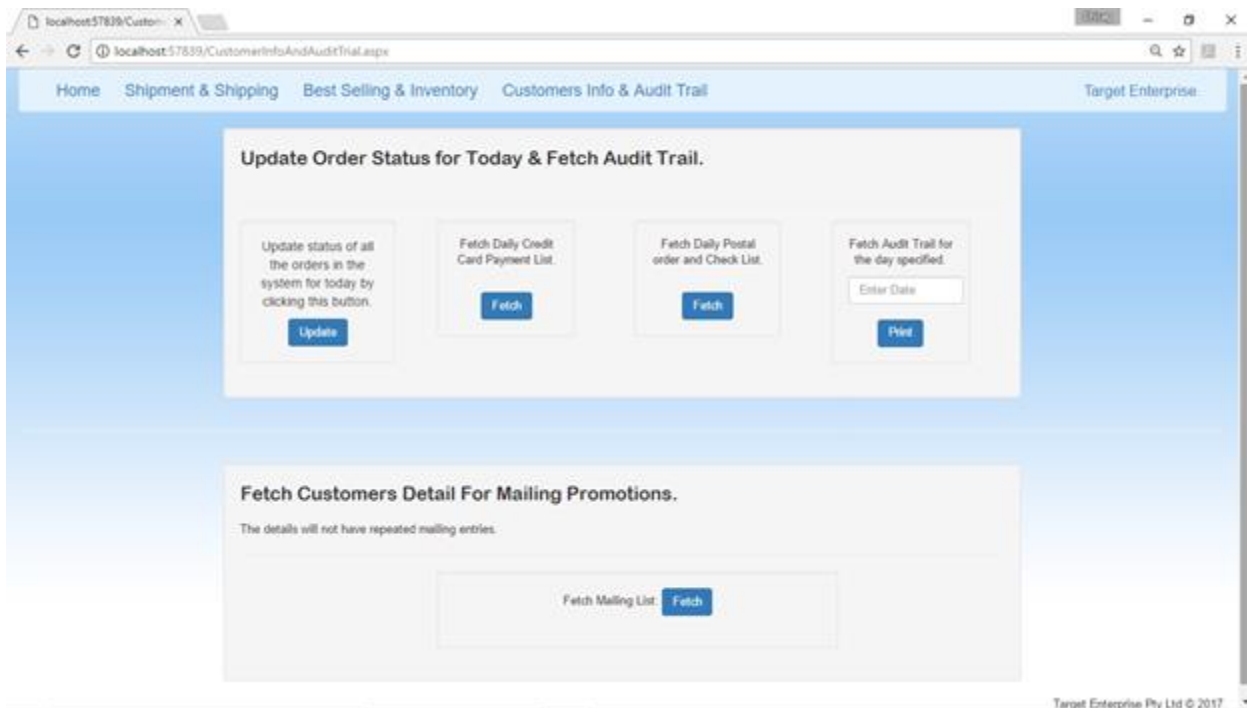


Image 9.5.1 Audit Trail

This page of the web application serves multiple and important purpose to the organization.

1. The update button in the update status part (top-left) will fire a query will change the pending Order status, so that it shouldn't be done manually for each of the orders in the system. The Admin can click this button at the end of each day, to continue proper flow every next morning. We can see the SQL query in Image 7.14
2. The Fetch button for the Fetch Daily credit card info will result in data of all the Order which were placed using Credit card for that day. As shown in the Image 7.2 above.
3. The Fetch button for postal order and checks will result in data of all orders which were placed using Postal Order and Check payments orders for that day. As shown in the Image 7.3 above.
4. The audit trail box will help the admin keep track of all the audits trail for any particular day by proving the date in the text box before submitting. This can be seen in the Images 7.4 and 7.5 above.

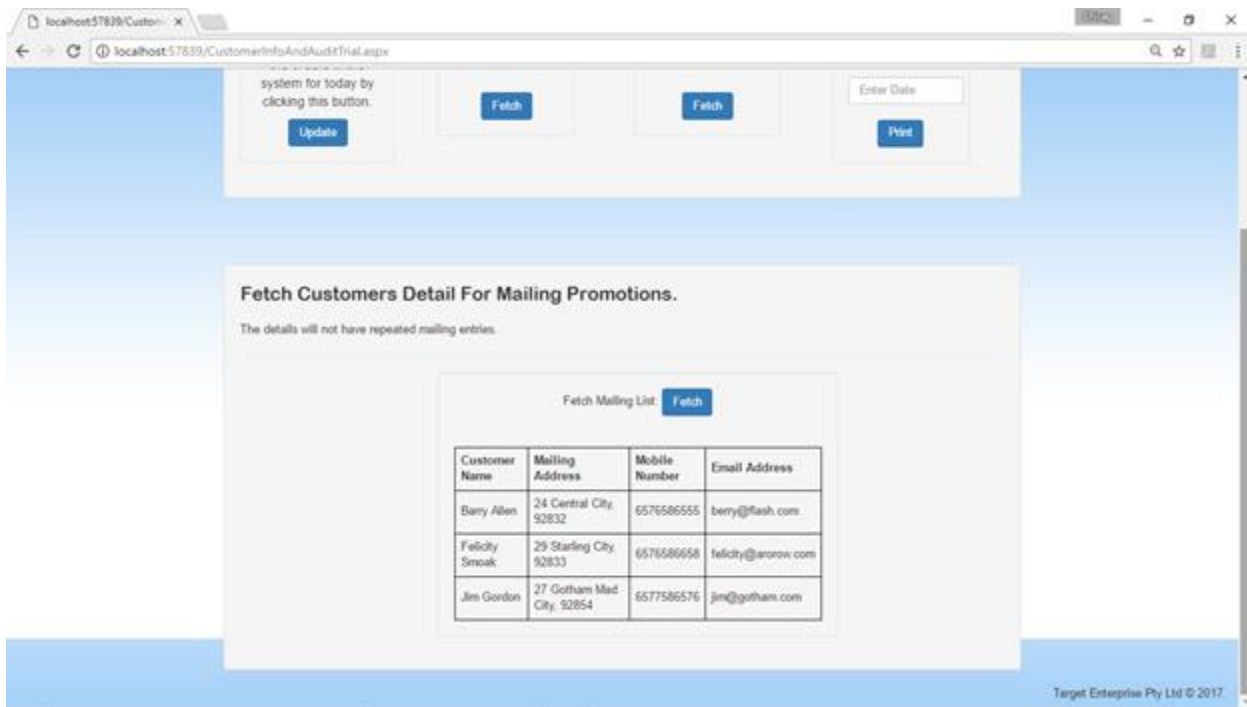


Image 9.5.2 Customer Info

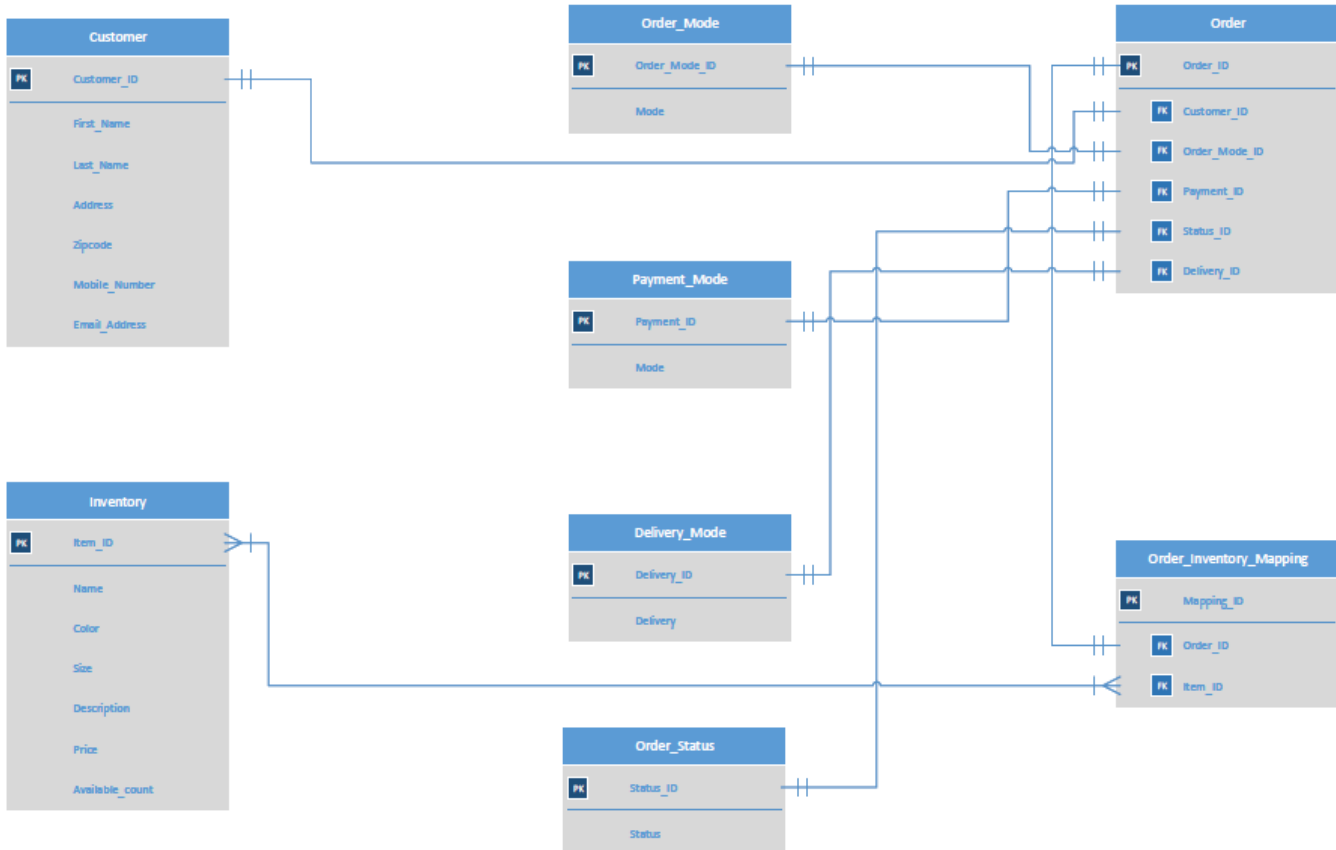
The web app image above shows the mailing list needed by the organization to avoid repetitive entries. By clicking the “Fetch” Button the system will display all the unique mailing address of customers from the database Customers table. This can be seen in the Image 7.9 above.

10. Conclusion

With the proposed database and the web application prototype, Target Enterprises (Pty) Ltd will be able to solve all the issues regarding dispatching and maintaining a customer database with their purchase history which will help them the company in maintaining an efficient tracking system. This will greatly assist the company in handling its project details, inventory and foster a good relationship among the different departments. The administrator will have a system which is improved, efficient and organised. For any organization to run smoothly, it is necessary that it utilizes its time efficiently. With this solution, the administrator at Target Enterprises (Pty) Ltd will definitely save time in the entire order management process and will be able to allocate time to successfully perform other managerial duties and responsibilities.

11. Appendices

Old Entity Relationship Database Model of Target Enterprise (Pty) Ltd.



Old Parenthetical method of the database system for Target Enterprise (Pty) Ltd.:

Order (Order_ID, Customer_ID, Order_Mode_ID, Payment_ID, Status_ID, Delivery_ID)

This table contains attributes of the order form which is filled by every customer. This is the main table of the ordering system, which contains foreign keys of other tables in the database.

Customer (Customer_ID, First_Name, Last_Name, Address, Zipcode, Mobile_Number, Email_Address)

This table contains the entry of every customer; this table will help the marketing department to avoid

duplicate entries of a single customer.

Order_Status (Status_ID, Status)

This table shows the status of the order; new entry can be added to this database easily and can be referred in the order table via Status_ID.

Order_Mode (Order_Mode_ID, Mode)

This table stores the order mode of the order; new entry can be added to this database easily and can be referred in the order table via Order_Mode_ID.

Delivery_Mode (Delivery_ID, Delivery)

This table stores the delivery type of the order; new entry can be added to this database easily and can be referred in the order table via Delivery_ID.

Inventory (Item_ID, Name, Color, Size, Description, Price, Available_Count)

This table stores all the items and every description of those items which can be ordered by the customers. This table can be used to keep track of the stocks coming in and going out as ordered.

Payment_Mode (Payment_ID, Mode)

This table stores the payment mode of the order; new entry can be added to this database easily and can be referred in the order table via Payment_ID.

Order_Inventory_Mapping (Mapping_ID, Order_ID, Item_ID)

This table stores data which helps in identifying the quantity of order of items on a single order form by mapping the Inventory and Order table.