Final Year Project Documentation

Data Mining in Grocery Stores

Project advisor: **Mr. Usama Riaz**

Project Manager: **Mr. Fahad Sabah**

Data Mining in Grocery Stores

Project Manager: Mr. Fahad Sabah

Signature: _____

Project advisor: Mr. Usama Riaz

Signature: _____

Presented by:

Roll#	Name:
14233	Muhammad Raheel Akram
14333	M Ijaz UL Rehman
14287	Bilal Hameed

Abstract:

Nowadays, massive amount of data has been generated from various sources, including industry, science and internet. As the amount of information grows exponentially, there is need to efficiently process and extract valuable information using data mining technologies. The aim of Data Mining is to extract hidden, predictive and potentially useful patterns from large databases.[9] This subject is becoming a very active research area and many different methodologies have been produced to solve industrial and scientific problem.

The objective of this project is to research Association Rules Discovery field and describe the process of developing software application, which extracts “useful patterns” from large datasets using Apriori Algorithm. This paper discusses the software development process as well as theoretical aspects of the project.

Project Registration

Project ID (for office use)

Type (Nature of project) [ ] Development [ ] Research [ ] R&D

Project Group Members Sr.# Roll # Student Name CGPA Email ID Phone # Signature

(i)

(ii)

(iii)

Name & Signature of Advisor (If students are eligible for FYP)

Plagiarism Free Certificate

This is to certify that, I am _____ S/D/o _____, group leader of FYP under registration no _____ at Computer Science Department & Information Technology, Superior University, Lahore. I declare that my FYP proposal is checked by my supervisor and the similarity index is _____% that is less than 20%, an acceptable limit by HEC. Report is attached herewith as Appendix A.

Signature: _____

I _____ show my consent to supervise the project titled; _____
_____

which consists of above listed students as group members

Date & Signature: _____ Designation: _____

Approval of FYP Committee Committee Member 1: Name: _____ Designation: _____ [ ] Accepted [ ] * Deferred [ ] *Rejected Signature: _____ *Remarks: _____ Committee Member 2: Name: _____ Designation: _____ [ ] Accepted [ ] * Deferred [ ] *Rejected Signature: _____ *Remarks:
_____

FYP Manager: Fahad Sabah [ ] Accepted [ ] *Deferred [ ] *Rejected Signature:_____ *Remarks:
_____

SUBMISSION STATEMENT

Raheel Akram(BSCS-14233), Bilal Hameed (BSCS-14287), M Ijaz ur Rehman(BSCS-14333) have successfully complete the final project Report named is: **Grocery Store by Using Data Mining**, at the Faculty of CS/IT, Superior University Lahore Campus, to complete the requirement of the degree of **Bachelors in Computer Science**.

Management Committee

Department of CS and IT

Superior University, Lahore Campus

SUPERVISOR	**HOD**
Faculty of CS and IT	**Faculty of CS and IT**

PROOFREADING CERTIFICATE

It is to confirm that I have read the document carefully and watchfully. I am influenced that the resulting project report does not hold any spelling, punctuation or grammatical mistakes as such. All in all, I find this document well planned and I am making sure that its objectives have been effectively meet.

Name: _____

Superior University Lahore Campus

Acknowledgement

I have taken hard work in this project. Though, it would not have been probable lacking support and help of many persons and organization. I would like to gratitude to all of them.

First I would like to thanks my ALLAH TALA who help me and give me the ability to work and complete the project. Then I am highly indebted to (SUPERIOR UNIVERSITY LAHORE CAMPUS) for their help and management as well as for provide essential information concerning the project & also for their support in complete the development.

I would like to convey my parents & my team members for their teamwork and support which help me in complete of this development.

Table of Contents

Chapter 1

Introduction

Introduction to Data Mining

Data Mining and its relevant influential new technology with unlimited potential to support the mega-companies focuses on the most off significant information in their data warehouses. Data mining tools envisage upcoming trends and actions, enabling companies to take active and new-knowledge based results. Automatic and potential analysis through the data mining goes outside the analysis of the previous procedures provided by the reflective tools, result provision systems. New data-mining outfits can respond to professional questions, which have usually been resolved for a long period. They look at records for unseen templates and find analytical information that professionals may lose because it is out of their prospects.

Many corporations have previously collected and refined large amounts of information. Data mining skills can be quickly deployed on current hardware and software stages to make good value of existing properties, integrate with new goods and systems as they are put online. When distributed on high performance or similar processing clients or high-performance server computers, data mining tools can analyze maximum records to provide answers to questions such as "Which customers will probably answer my next promotional email and why?"

This white paper provides an overview to the basic mining data skills. Examples of low-cost applications show their importance to today's business location and basic explanation of data warehouse designs can change to provide data extraction values to end users.

Foundation of Data Mining:

D-M's methods are the result of long procedure of product enquiries and improvement. This progress initiated when business data, initially stored, continuous to increase data access, and newly, created technologies that allow workers to browse their data. D-M is prepared for application in the corporate community because it is maintained by three mature technologies:

• Mass data collection

• Powerful multiprocessor computers

• Data mining algorithms

Scope of Data Mining:

Data mining owes its term to the likenesses among searching for precious business data in a huge database, such as looking for gbs of scanner store data and removing a mountain from a precious mineral vein. Both procedures need to sieve through a large quantity of material or intelligently probe to catch exactly where the value exist in. By providing satisfactory size and quality databases, data mining skill can create new business occasions by providing these features:

Automate the Prediction of Trends and behaviors:

Data mining systematizes the search method for projecting information in huge databases. Questions that usually require general analytical work can currently answered directly from the data quickly. The example of a analytical problem is targeted marketing. Data mining usages the data for previous advertising emails to recognize goals that maximize return on asset in future e-mail. Other analytical issues include impoverishment prediction and other default forms and sections of a population that can return similarly to certain events.

Automates the discovery and unknown patterns:

Data mining outfits cross the databases and find the earlier hidden templates in one step. An example of detection of the models is the investigation of retail sales data to find apparently dissimilar products frequently purchased together. Other model finds issues contain detecting fake credit card transactions and finding abnormal data that may be data input errors.

Data mining skills can create the profits of automation on surviving hardware and software stages and can be executed on new systems as surviving stages are upgraded and new products are developed. When data mining outfits are deployed on high performance similar processing systems, they can scan mass databases in notes. Quicker processing allows users to habitually experiment with multiple templates to recognize difficult data. High speed makes it handy for users to evaluate huge amounts of data. Larger databases, in turn, produce better calculations.

Most Commonly Databases used in Data mining are:

• Artificial neural networks: nonlinear analytical models that pick up through training and look like biological neural networks in the structure.

• Decision trees: tree configurations representing groups of results. These decisions create rules for classifying a set of data. Decision particular metadata includes grouping and regression trees (carts) and Chi Square collaboration detection (CHAID).

• Genetic algorithms: Optimization methods that use procedures such as genetic grouping, mutation, and normal selection in a project built on evolutionary concepts.

• Closest approach: a skill that categorizes each record in a data set based on a grouping of k class classes like to it in a set of old data (where k ³ 1). Sometimes the closest technique of k-neighbor is called.

• Rule induction: mining of if-then useful rules from data based on arithmetical implication.

Many of these technologies have been used for more than a time in specific analysis outfits that operate with comparatively minor volumes of data. These features are now changing to integrate seamlessly with the usual data warehouse and OLAP platforms. The addition to this white paper offers a glossary of mining terms.

How Data Mining Works:

How exactly can data mining tell you main things you did not recognize or what would occur next? The method used to execute these tasks in extracting data is called modeling. Modeling is simply the act of construction a model in a condition where you know the answer and then applies to another condition you do not have. For example, if you were watching for a Spanish galleon sunk on the high seas, the first object you can do is to study the times when the Spanish paragon had been creating by others in the past. It can be noted that these ships are often found on the banks of the Bermuda and that are some features of the sea currents and some paths probably occupied by the ship's captains at that time. You notice these parallels and build a pattern that take in features that are mutual to the positions of these lower resources. With these models in your influence you move in search of a paragon where your model specifies that it is very likely that such a condition has been given in the past. We courage that if you have a good model, you will discover your treasure.

This modeling act is rather that people are liability a long time, surely already the advent of computers or data mining technology. However, what occurs in computers is not very altered from how folks create models. The computers are weighted down with a lot of data on a variety of conditions where a response is identified and therefore the data mining software on the computer must ride such data and distil the data features that must be included in the model. Once the model has been constructed, it can be used in parallel conditions where the answer is unidentified. For example, suppose you are marketing manager of a telecommunications corporation and want to obtain some long-distance telephone clients. He might be simply go out and casually send the vows to the over-all people just as he could casually traverse through the seas in search of sunken materials. In either case it would not have reached the results it wanted, and of sequence it would have the chance to do abundant well than at unplanned; you might be use your business practice stored in your database to construct a template.

As an advertising manager, you have right of entry to a lot of info about all of your customers - your age, sex, credit account, and the use of long space calls. The good news is that you also have a portion of material about your prospects: your age, sex, your credit history, and so on. Your badly-behaved is that you do not be familiar with the use of long distance calls from these potential customers (since you're probably the clients of your competitors). It would similar to focus on predictions that have huge amounts of long distance use. You can do this by structure a model. Table 1 shows the data used to build a prospect ideal for new customers in a data storeroom.

	Prospects	Customers
Proprietary Figures (Customers Transactions etc.)	Target	Unknown
General Information (Demographic information etc.)	Unknown	Unknown

Table 1 – Data Mining for Prospecting.

The purpose of the survey is to create some designed hypotheses about the info in the minor right quadrant based on the template created, from Customer Overview to Customer's Property Figures. For example, a modest model for a telecom company could be: 97% of my clients earning more than $ 70,000 / year spends extra $ 90 / month on long distance. This model might be then applied to

outlook data to try to say a little about patented data that this telecom companies currently do not have access.

Marketing Testing is a great cause of data for this type of model. Extracting the outcomes of a test market that represents a large but comparatively minor sample of views can provide a basis for identifying good views in the general market. Table-2 displays an-other mutual scenarios for building models: to guess, what will occur in the upcoming days.

Table 2 – Data Mining for predictions.

	Yesterday	Today	Tomorrow
Dynamic Data (Customers Transactions etc.)	Unknown	Unknown	Unknown
Static Info and Current Strategies (Demographic data, Marketing plans etc.)	Unknown	Unknown	Unknown

Architecture for Data Mining:

To better apply these progressive techniques, they must be completely combined with a data warehouse as well as cooperating and stretchy business analysis tools. Several data mining tools presently work outdoor the warehouse, requiring additional procedures for removing, importing and studying data. Additionally, when new knowledge requires an working application, addition with the warehouse simplifies the application of data extraction results. The resulting analytical store can be applied to increase business procedures across the institute in areas such as promotional campaign management, scam detection, new product launch, and so on. Figure 1 explains an advanced analysis architecture in a large data storeroom.

The perfect initial point is a data warehouse that contains a mixture of interior data that tracks all client contacts along with outside market data on the motion of the competition. Basic cable information also provides a good base for searching. This store can be deployed in several relational database systems: Sybase, Oracle, Redbrick, etc., and it needs to be enhanced for stretchy and fast access to data.

An online analytics server (OLAP) allows you to apply a classier to end-user business-model, when exploring the data storeroom. Multidimensional assemblies let the user to evaluate the data when they want to see their business, brief the invention line, area and other key prospects of their business. The Data Mining Waiter must be combined with the data storeroom and the OLAP server to integrate ROI based business investigation directly into this structure. A progressive, process-centered metadata model describes the mining targets for specific business problems, such as operation management, search and optimization of promotions. Addition with the data storeroom allows you to right implement and monitor operational decisions. As the storeroom produces with new conclusions and results, organization can frequently extract best practices, to relate them to future results.

This design signifies important change from straight decision support systems. Instead of only providing end-user data using query and reporting software, Progressive Analysis Server smears user business replicas directly to the store and returns an active study of the most related info. These results improve metadata on the OLAP server by providing a layer of active metadata that signifies a purified view of the data. Reports, meditation and other analytical tools can be applied to plan future actions and check the effect of such plans.

Profitable applications:

An eclectic choice of companies has implemented positive data mining applications. Although the first users of this technology belonged to areas requiring much information, such as economic services and shortest marketing, technology is relevant to any company that wants to use a large data storeroom to well manage their customer relations. Two serious issues for the achievement of the data mining business are: a large and well-integrated data storeroom and a well-defined sympathetic of the business process where data mining (such as customer prospecting, storage, management of data mining campaigns, etc.).

Here some successful Applications areas includes:

• A medical company can examine their current sales activities and results to increase the selection of high-quality surgeons and determine which marketing actions will have the greatest effect in the upcoming months. The data should contain competitive market activity as well as data on native health systems. The results can be dispersed to the sales force through a large link of areas that permits agents to review approvals from the outlook of key attributes in decision making. The

endless and active analysis of the data store room agrees you to apply the best practices of the entire organization to definite sales conditions.

• A credit card corporation can leverage its large customer data transaction archive to find customers who might be attracted in a new credit product. By a small test mail, you can recognize customer qualities with attraction for the product. Recent projects have shown a reduction of more than 20 times the cost of targeted mail campaigns compared to conservative methodologies.

• An expanded transport company through strong straight sales force can apply data mining to recognize the best predictions for its services. With data mining to study its customer experience, this company can create a unique dissection that identifies high-value perspective attributes. Applying this separation to overall business database such as those providing by Dun & Bradstreet can generate a list of possible priorities per region.

• A huge consumer things business can apply data mining to recover its retail process. Consumer Panel Data, Shipment, and Competitive Action can be applied to recognize the reasons for brand and store change. Concluded this analysis, the producer can select the promotion plans that best reach the mark customer segments.

Respectively these examples have a pure mutual approach. Take advantage of implicit customer knowledge in a data warehouse to decrease costs and expand the value of customer relations.

Chapter 2

Glossary Data Mining Terms

- Artificial Neural Network Nonlinear analytical models that study through training and look like biological neural networks in the structure.
- CART Classification and regression of trees. A resolution tree method used to classify a data set. Delivers a set of rules that you can apply to a new data set (unclassified) to guess which records will have a certain result. Segment a data set by making bidirectional partitions. It needs less data preparation than CHAID.
- CHAID Chi Square communication finding. A conclusion tree technique used to classify a data set. It delivers a set of rules that can be applied to a new data set (unclassified) to guess which records will have a certain result. Sections a data set using chi-square tests to make multidirectional divisions. Preceded, and requires more data preparation than CART.
- Anomalous Data, Data that results from faults (Data Entry Errors) or that represent uncommon events. The abnormal data should be wisely considered as it may contain significant info.
- Classification The process of sharing a set of data into groups exclusively so that the members of individually are as close as the others and that the different groups are as far away as likely, where the distance is dignified with respect to the variables that you are trying to predict. For example, a distinctive classification problem is to divide a database of companies into groups as same as possible with respect to a credit adaptable with "Good" and "Poor" values.
- Analytical Model, A configuration and process for evaluating a data set. For example, a decision tree is a model for sorting a data set.
- Data Mining, the Extract hidden prediction data from huge databases.
- Data Navigation The procedure of presenting different sizes, sectors, and detail levels of a multidimensional database.
- Data Cleansing The process of confirming that all values of a data set are reliable and properly recorded.
- Clustering is the process of separating a set of data into groups entirely so that the members of each group are as close as the others and that the dissimilar groups are as far away as possible, where distance is dignified with respect to all existing variables

- Dimension, in a flat or relational database, each field in a record signifies a dimension. In a multidimensional database, a dimension is a set of analogous entities; For example, a multidimensional sales database may contain Product, Time, and City dimensions.
- Decision tree is a tree construction representing a set of choices. These decisions create rules for classifying a set of data
- Data Visualization Visual clarification of compound relationships in multidimensional data.
- Data Warehouse is a system for storing and distributing massive data.
- Genetic Algorithm is Optimization techniques that use procedures such as genetic grouping, alteration and natural selection in a project based on the concepts of natural evolution.
- Linear Model is the Analytical model that accepts linear-relationships in the constants of the variables studied.
- Linear Regression is arithmetical technique used to discover the best linear-relationship among objective variable (employee) and its predictors.
- Logistic Regression is a linear regression that contains the extents of a categorical target variable, such as the type of client, in a population.
- Multidimensional Database is a db that designed for online analytics. Organized as a multidimensional hypercube with an affiliation per dimension.
- Multiprocessor computer is a computer that contains multiple processors linked to a network.
- Non-linear Model is a model that does not accept linear relationships in the factors of the variables studied.
- Nearest Neighbor method that classifies each record in a data set created on a combination of k-class classes more similar to it in a set of historical data (where k ³ 1). Sometimes it is called closer to k-neighbor technique.
- OLAP is analytical online processing. It refers to array based database applications that agree users to view, traverse, operate, and study multidimensional databases.
- Parallel Processing is coordinated use of multiple processors to perform computational tasks. Parallel processing can take place on a multiprocessor computer or on a workstation or PC network.
- Outlier is the element of a data whose value is outside the limits of most other sample values. It may specify abnormal data. It should be carefully observed. They can convey important information.
- RAID Means Reducing Array of Low-cost disks. A technology for effective parallel data storage for high performance computing systems.

- Predictive Model is a model and a processor for predict the values and specifies the variables in database.
- Rule Induction is the if-than valuable rules form data based on numerical significance.
- SMP means Symmetric Multiprocessor in which memory is shared between processors.it is the type of multiprocessor computers.
- Terabytes One trillion bytes.
- Retrospective data analysis, the type of data study in which deliver the insight into trends, behaviors or events that already happens.
- Time Series Analysis the Analysis of a sequence of measurements carried out at specific time intervals. Time is often the dominant measurement of data.

Chapter 3

Apriori-Algorithm v/s FP-Growth-Algorithm for Frequent Item Set Mining

Comparison between Apriori and FP-Growth algorithms

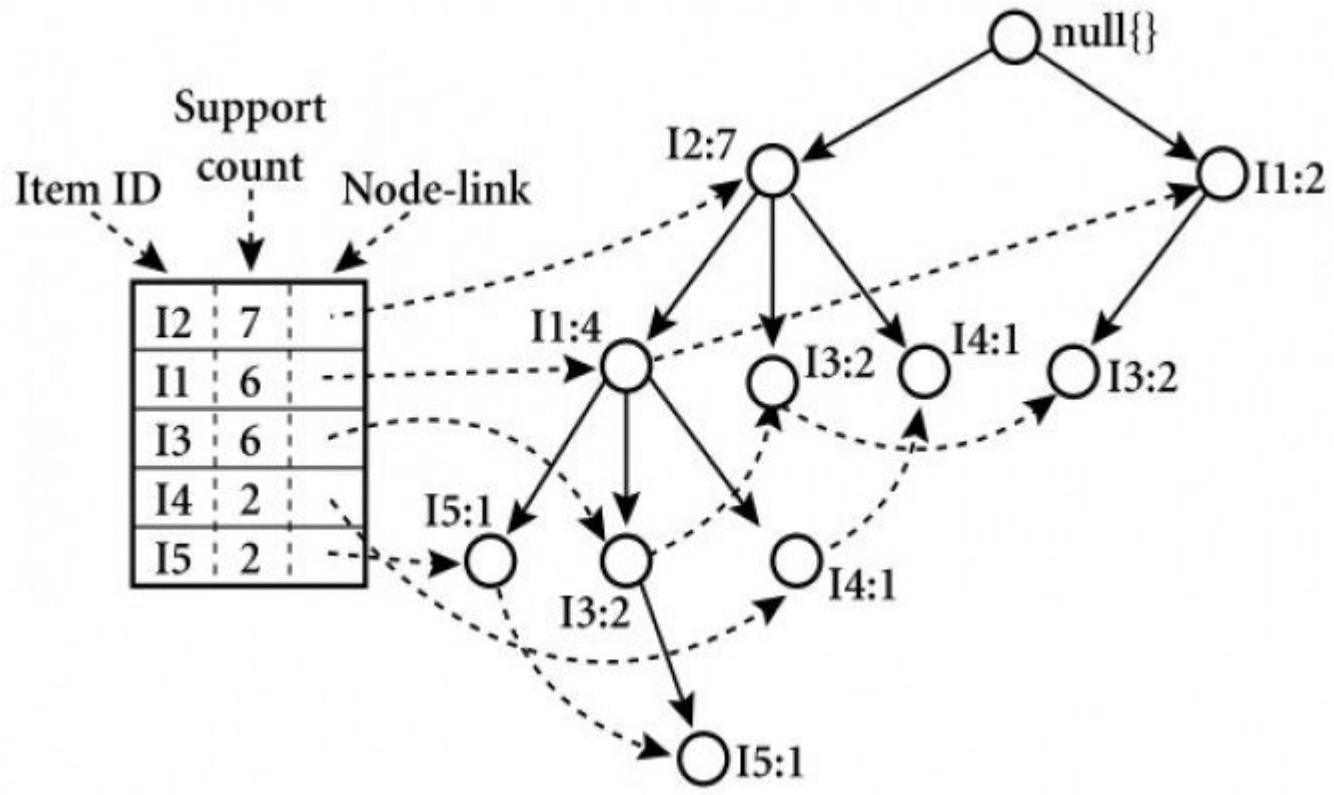


Figure. 1

Frequent Item Set mining is a necessary part of many M-Learning algorithms. What this technique is intended to do is to extract the most frequent, largest and biggest items list of transactions containing every or several items each. For example:

Suppose (*T)* is a list of n transactions [t_1, t_2, ..., t_n]. Every transaction t_i contains a list of k_t items [a_{i1}, a_{2i}, ..., a_{ik}].

So here T = [t_1 = [a_{11},a_{12}, ... A_{1k}],,t_2 =[a_{21},a_{22}, ... A_{2k}],..., t_k = [a_{t1}, a_{t2}, ... A_{tk}]].

Frequent Item Set for T is FIS_T, where $\forall s \in FIS_T$, s is the most frequent and largest set of items, which:

- Is not contained within another set in FIS_T.
- Appears at least m times (m is a number as the minimum support threshold).

Example of this FIS technique, a customer C1, with a minimum support threshold m of 50 % and we want to mine the FIS for C1.

T_{C1} = [[beer,wine,rum],[beer,rum,vodka],[beer,vodka],[beer,wine,rum]].

$| T_{C1} |$ = 4

M = 2 (50 % of 4)

FIS_{C1} = { {beer,wine,rum},{beer,vodka}}

From this example, {beer,wine,rum} and {beer,vodka} appear in 2 of the transactions.

Any one may be asking why isn't {beer,rum} inside FIS_{C1}, if it appears on 3 times. Answer is due to {beer,rum} already contained inside {beer,wine,rum}, which is larger.

This is not a concrete implementation algorithms, now explanation of implementations of:

Apriori and FP-Growth.

Apriori Algorithm

Apriori is based on the fact that if a subset S appears k times, any other subset S' that contains S will **k times or less**. So, if S doesn't pass the minimum support threshold, **neither does S'**. There is **no need to calculate S'**, it is discarded **a priori**.

Now we're going to show you an example of this algorithm.

Let's suppose a client Mario with transactions [[beer,wine,rum], [beer,rum,vodka], [beer, vodka], [beer,wine, rum]], and a minimum support threshold m of 50 % (2 transactions).

First step: Count the singletons and apply threshold

The singletons for Mario are:
beer: four,
wine: two,
rum: three,
vodka: two

All of the items appear m or more times, so no one of them will discarded.

2nd step: Generating pairs, count them and applying threshold

Pairs created were: {beer,wine}, {beer,rum}, {beer,vodka}, {wine,rum}, {wine,vodka}, {rum,vodka} }. Now proceed to count them and applying the threshold.

{beer, wine}: two
{beer, rum}: three
{beer, vodka}: two
{wine, rum}: two
{wine, vodka}: zero
{rum, vodka}: one

{wine,vodka} and {rum, vodka} cannot passed the threshold, so these are discarded & any other sub-combination both of them can generate.

Remaining pairs will put into temporal association set.
Assoc = {{beer,wine}, {beer,rum}, {beer,vodka}, {wine,rum} }

Step-N: Generating triplets, quadruplets and etc., count and apply threshold & remove containing item sets.

Generate triplets from pairs.
Triplets = {{beer, wine,rum}, {beer,wine, vodka}, {beer,rum, vodka}, {wine, rum,vodka} }

Now count them:
{beer,wine,rum}: two
{beer,wine, vodka}: Zero
{beer, rum, vodka}: One
{wine, rum,vodka}: Zero

Only {beer, wine, rum} passed the threshold, so now proceed to add it to Assoc, but first, removing the subsets that {beer, wine, rum} contains.

Before adding remaining triplet Assoc looked like this: { {beer,wine}, {beer,rum}, {beer vodka}, {wine, rum } }.

Adding the triplet, and removing the subsets that are inside {beer, wine}, {beer, rum} and {wine, rum} are the ones that should go.

Assoc now it seem like { {beer, wine,rum}, {beer,vodka} }, and this is **final result**.

If more than one triplet after aplying the threshold, proceed to generat the quadruplets, count them, aplying the thresshold, adding them to assoc and remove the subsets that each quadruplet contains.

Disadvantages of Apriori:

- The generation of candidates could slow (pairs, triplets, etc.).
- The generation of candidate could generate multiple duplicate depends on the implementation.
- The method of counting iterates through all of the transactions each time.
- Constant items make the algorithm a lot heavier.
- Memory consumption is huge

Advantages of Apriori

Apriori calculates more sets of frequent items.

FP-Growth

FP-Growth is improvement of apriori designe to eliminate some of the heavy works in apriori. This was planned with the benefit of map-Reduce taken into account, so it works well with any distributed system focused on map-Reduce. It simplifies all the problems in apriori by using a structure called an FP-Tree. In an FP-Tree each node represents an item and its current count, and each branch represents a diferent assoc.

Algorithm is divided in 5 steps. Here a simple example:

A Client is named C1 and here his transactions:
TC1= [[beer, bread, butter,milk] , [beer,milk, butter], [beer,milk,cheese], [beer, butter,diapers,cheese], [beer, cheese, bread]]

Step-1:

First step is counting all the items in all the transactions
TC1= [beer: five, bread: two, butter: three, milk: three, cheese:three, diapers: one]

Step 2:

Next applying threshold set previously. For this example let's say a threshold of 30% so each item has to appear at least 2 time.

TC1= [beer: five, bread:two2, butter:three, milk: three, cheese: three, ~~diapers: 1~~]

Step 3:

Now sorting list according to the count of each item.

$TC1_{Sorted}$ = [beer: five, butter: three, milk:three, cheese: three, bread: 2]

Step 4:

Now building the tree. Go through each of the transactions and add all the items in the order they appear in sorted list.

Transaction add= [beer,bread,butter, milk]

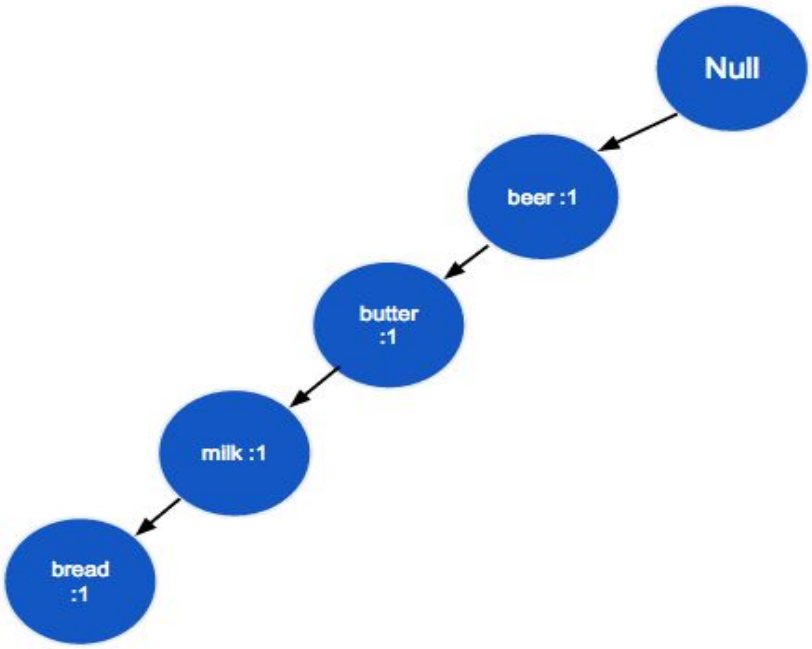


Figure no. 2

Transaction 2: [beer milk, butter]

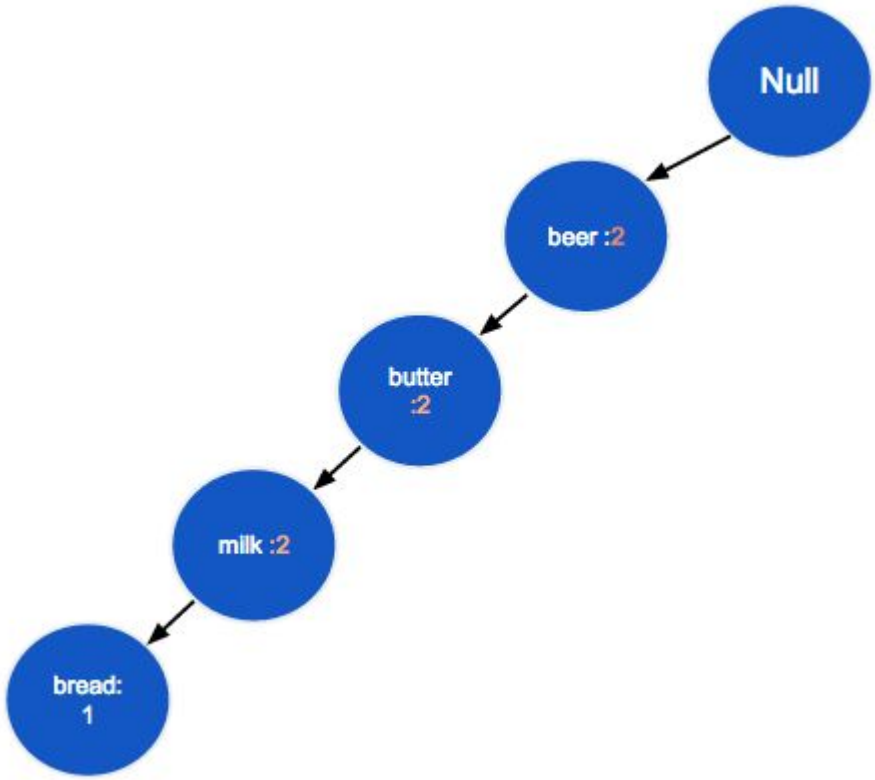


Figure no. 3

Transaction 3=[beer, milk, chese]

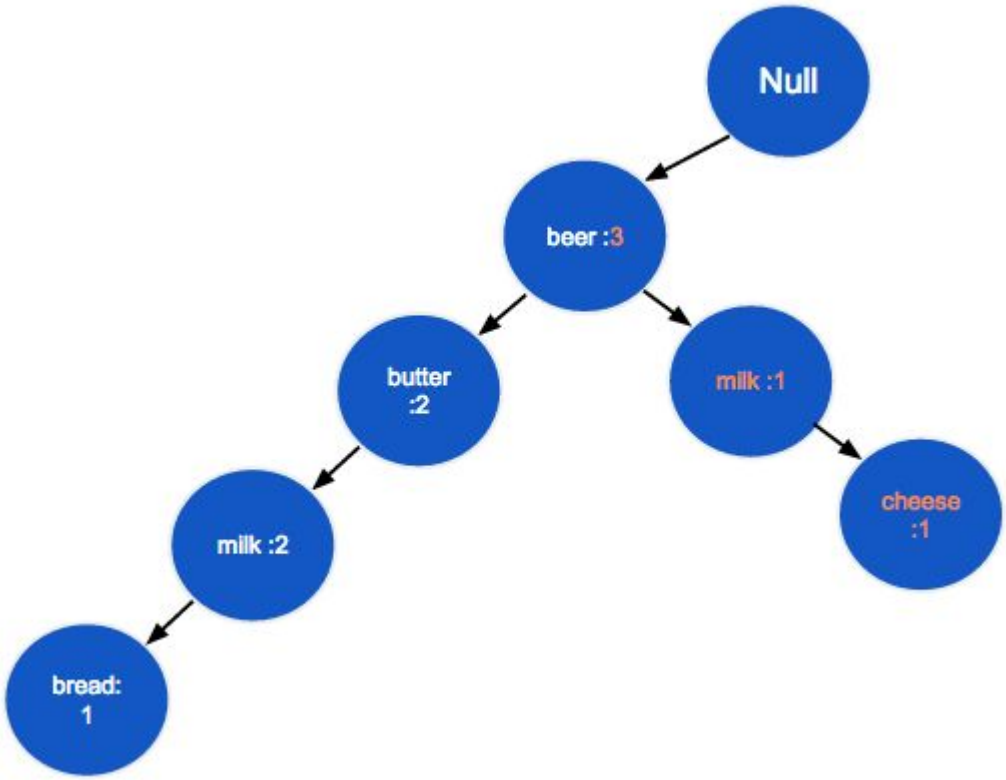


Figure no. 4

Transaction 4= [beer , cheese bread]

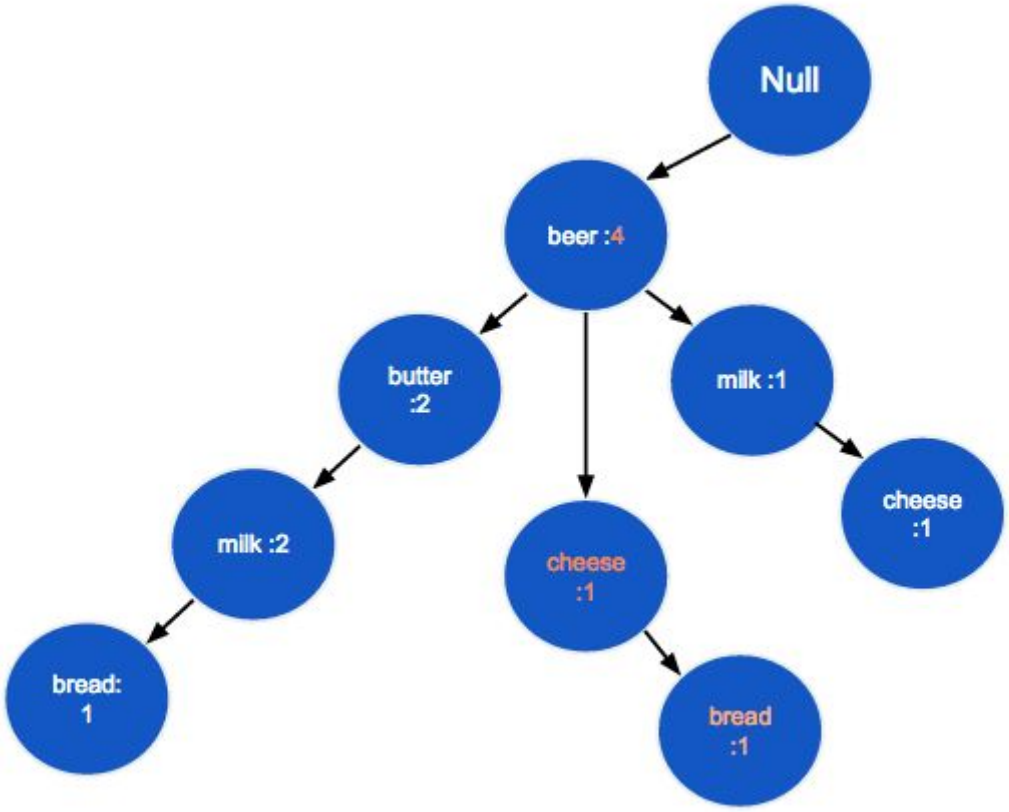


Figure no. 5

Transaction 5 = [beer, cheese, diapers]

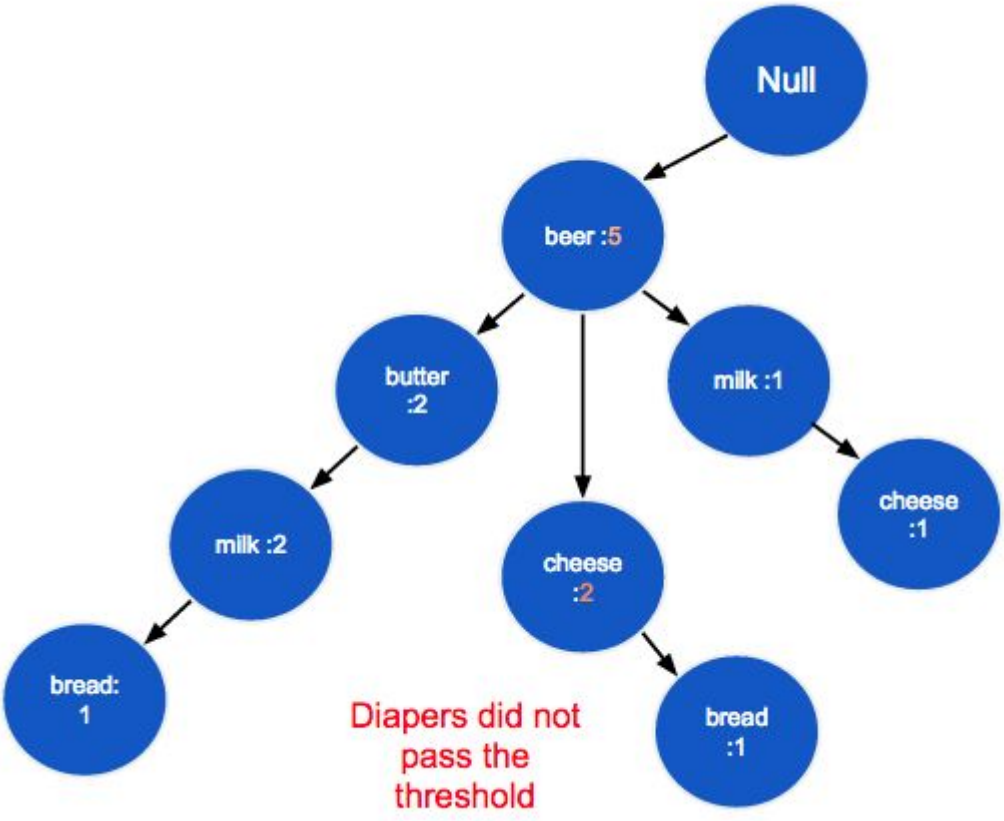


Figure no. 6

Step 5:

In order to get the assoc now go through every branch of the tree and only including in the assoc all the nodes whose count passed the threshold.

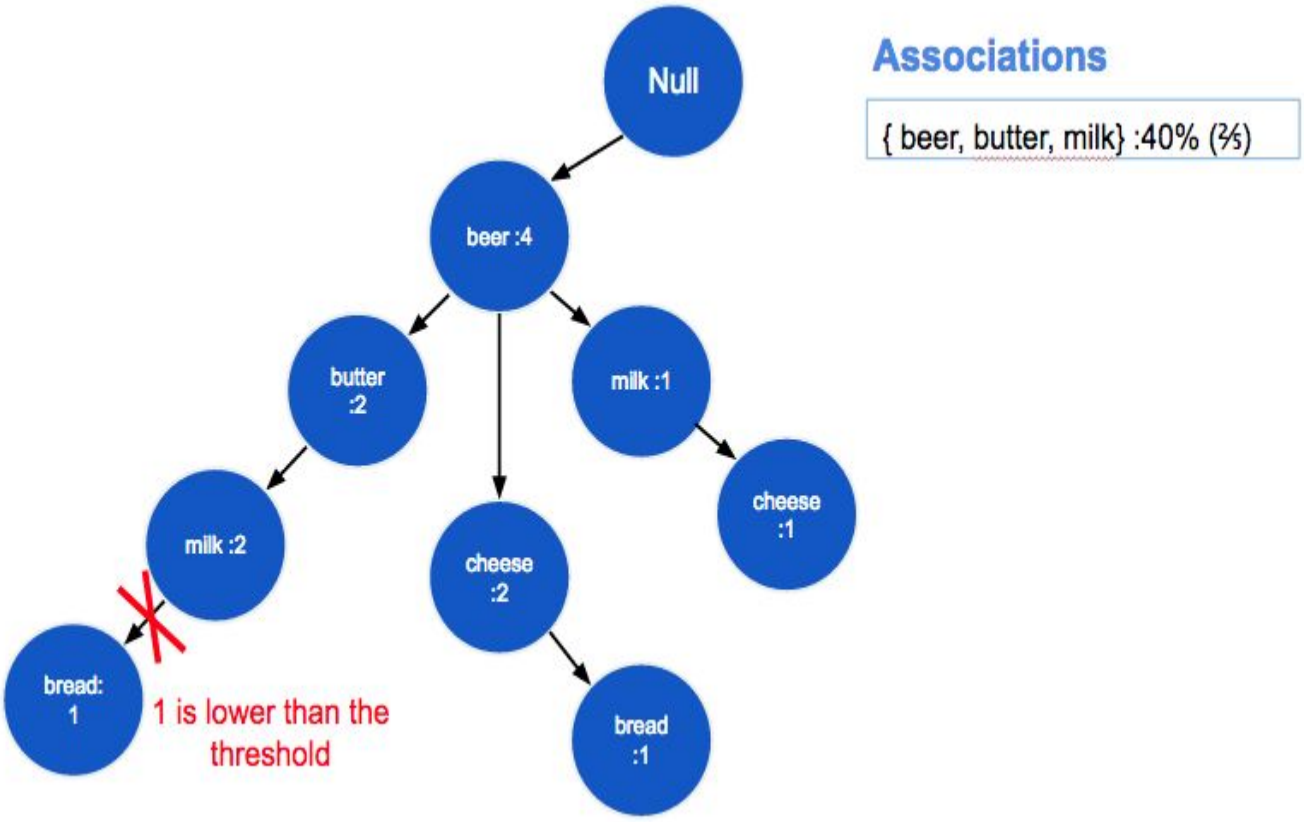


Figure no. 7

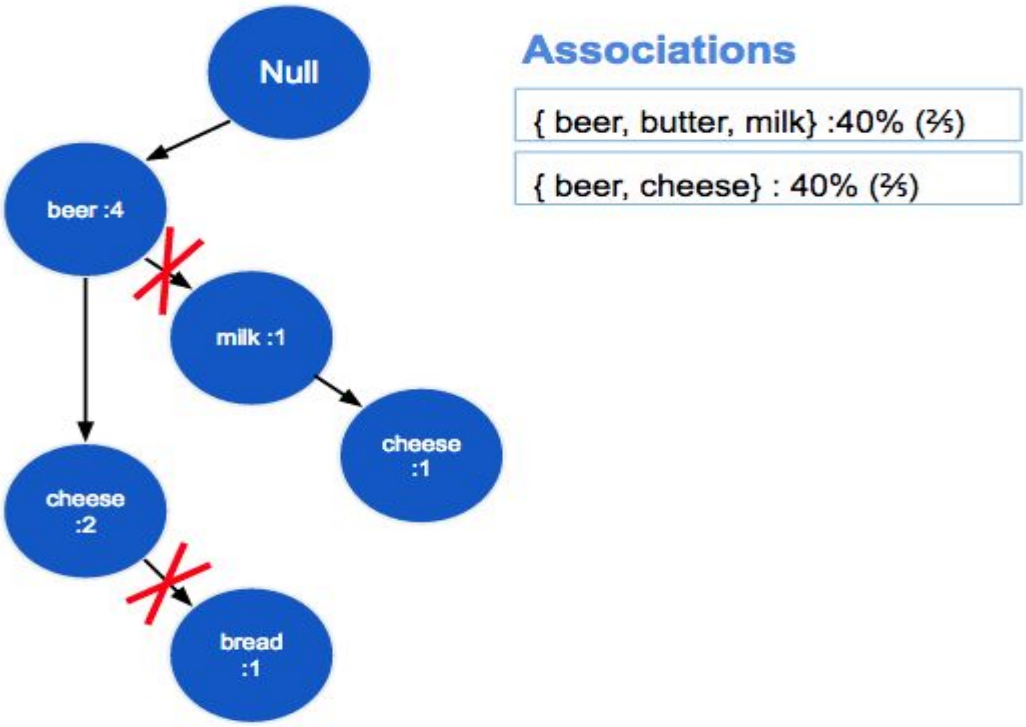


Figure no. 8

FP-Growth Biggest Advantages

Big advantage found in FP-Growth is the fact that the this only needs to read the file 2 times, as opposed to apriori who reads it once for every iteration.

Another big advantage is that it removes need to calculate the pairs to be counted, which make very processing heavy, because it use the FP-Tree. This makes it O(n) which is much quicker than apriori.

The FP-Growth stores in memory a compact version of the DB.

FP-Growth Bottlenecks

The big problem is the interdependency of data. Problem is that for the parallelization of the algorithm some that still needs to be shared, which creates a bottleneck in the shared memory.

Apriori vs FP-Growth

Algorithm	Technique	Runtime	Memory usage	Parallelizability
Apriori	Generate singletons, pairs, triplets, etc.	Candidate generation is extremely slow. Runtime increases exponentially depending on the number of different items.	Saves singletons, pairs, triplets, etc.	Candidate generation is very parallelizable
FP-Growth	Insert sorted items by frequency into a pattern tree	Runtime increases linearly, depending on the number of transactions and items	Stores a compact version of the database.	Data are very inter dependent, each node needs the root.

Table no. 3

Chapter 4

Project Introduction

Overview

This chapter gives an introduction of the project, describing the main goals and objective to be achieved. Moreover, it shows the outline of the project, briefly describing each part.

Introduction

Data mining is a powerful method with great potential to help companies focus on the most important information in the data they have collected about the behavior of their customers and potential customers. It discovers information within the data that queries and reports can't effectively reveal.

Objective

“We want to provide an intelligent software that helps to the grocery store owners. In this software we will provide visual display of the items of grocery store, and suggest that how they can arrange the items for better sales.”

Problem Description

People are facing difficulties when they buy anything from a grocery store. It’s hard to find everything they need, because items are not in order. Stores don’t know their target market and which customers wants to purchase what kind of goods and things.

Methodology

For this project we will use some analytical data mining technique to find frequent data set of grocery stores and make predictions to take some decisions. This will also classify data of grocery store.

Scope

It is important for the grocery stores to arrange their items which are east to access by customers. It will help and save the time when they are arranging data set. They can increase their sale through this data mining software. All types of grocery stores will be the target market.

Feasibility Study

Generally, there is no risk involved in our project. For this project the main required item is data set.

Solution Application Areas

This application areas are grocery stores. It can work in every kind of grocery store, like Hyper Star, Metro and Emporium Mall etc. But we are targeting the local grocery stores.

Tools/Technology

Possible list of tools and technologies will be used in the completion of project

- Asp.net
- SQL server
- Weka
- R-Studio
- Rapid Miner

User interfaces

All pages of the system are following a consistent theme and clear structure. The occurrence of errors should be minimized through the use of radio buttons and scroll down in order to reduce the amount of text input from user.

Client Side

The system is a web based application; clients are requiring using a modern web browser such as Mozilla Firebox 1.5, Internet Explorer 6 and Enable Cookies. The computer must have an Internet connection in order to be able to access the system

Server Side

An IIS Web server will accept all requests from the client and forward specific requests to Database. A development database will be hosted locally (using SQL); the production database is hosted centrally (using SQL Server).

Client Side

An OS is capable of running a modern web browser which supports HTML version 3.2 or higher.

Communications interfaces

The HTTP protocol will be used to facilitate communications between the client and server.

Site adaptation requirements

There should no site adaptation requirement since the Web Application Server was setup and running IIS web application.

Chapter 5

Testing

Software Testing - Overview

Testing:

Software testing is a process of verifying and validating that a software application or program

1. Meets the business and technical requirements that guided its design and development, and

2. Works as expected.

Software testing also identifies important defects, flaws, or errors in the application code that must be fixed. The modifier “important” in the previous sentence is, well, important because defects must be categorized by severity (more on this later).

During test planning we decide what an important defect is by reviewing the requirements and design documents with an eye towards answering the question “Important to whom?” Generally speaking, an important defect is one that from the customer’s perspective affects the usability or functionality of the application. Using colors for a traffic lighting scheme in a desktop dashboard may be a no-brainer during requirements definition and easily implemented during development but in fact may not be entirely workable if during testing we discover that the primary business sponsor

is color blind. Suddenly, it becomes an important defect. (About 8% of men and .4% of women have some form of color blindness.)

The quality assurance aspect of software development—documenting the degree to which the developers followed corporate standard processes or best practices—is not addressed in this paper because assuring quality is not a responsibility of the testing team. The testing team cannot improve quality; they can only measure it, although it can be argued that doing things like designing tests before coding begins will improve quality because the coders can then use that information while thinking about their designs and during coding and debugging. Software testing has three main purposes: verification, validation, and defect finding.

The verification process confirms that the software meets its technical specifications. A "specification" is a description of a function in terms of a measurable output value given a specific input value under specific preconditions. A simple specification may be along the line of "a SQL query retrieving data for a single account against the multi-month account-summary table must return these eight fields <list> ordered by month within 3 seconds of submission." The validation process confirms that the software meets the business requirements. A simple example of a business requirement is "After choosing a branch office name, information about the branch's customer account managers will appear in a new window. The window will present manager identification and summary information about each manager's customer base: <list of data elements>." Other requirements provide details on how the data will be summarized, formatted and displayed. A defect is a variance between the expected and actual result. The defect's ultimate source may be traced to a fault introduced in the specification, design, or development (coding) phases.

What is Testing?

Testing is the process of evaluating a system or its component(s) with the intent to find whether it satisfies the specified requirements or not. In simple words, testing is executing a system in order to identify any gaps, errors, or missing requirements in contrary to the actual requirements.

According to ANSI/IEEE 1059 standard, Testing can be defined as - A process of analyzing a software item to detect the differences between existing and required conditions (that is defects/errors/bugs) and to evaluate the features of the software item.

Who does Testing?

It depends on the process and the associated stakeholders of the project(s). In the IT industry, large companies have a team with responsibilities to evaluate the developed software in context of the given requirements. Moreover, developers also conduct testing which is called **Unit Testing**. In most cases, the following professionals are involved in testing a system within their respective capacities:

- Software Tester
- Software Developer
- Project Lead/Manager
- End User

Different companies have different designations for people who test the software on the basis of their experience and knowledge such as Software Tester, Software Quality Assurance Engineer, QA Analyst, etc.

It is not possible to test the software at any time during its cycle. The next two sections state when testing should be started and when to end it during the SDLC.

When to Start Testing?

An early start to testing reduces the cost and time to rework and produce error-free software that is delivered to the client. However, in Software Development Life Cycle (SDLC), testing can be started from the Requirements Gathering phase and continued till the deployment of the software. It also depends on the development model that is being used. For example, in the Waterfall model, formal testing is conducted in the testing phase; but in the incremental model, testing is performed at the end of every increment/iteration and the whole application is tested at the end.

Testing is done in different forms at every phase of SDLC:

- During the requirement gathering phase, the analysis and verification of requirements are also considered as testing.
- Reviewing the design in the design phase with the intent to improve the design is also considered as testing.
- Testing performed by a developer on completion of the code is also categorized as testing.

When to Stop Testing?

It is difficult to determine when to stop testing, as testing is a never-ending process and no one can claim that a software is 100% tested. The following aspects are to be considered for stopping the testing process:

- Testing Deadlines
- Completion of test case execution
- Completion of functional and code coverage to a certain point
- Bug rate falls below a certain level and no high-priority bugs are identified
- Management decision

Verification & Validation

These two terms are very confusing for most people, who use them interchangeably. The following table highlights the differences between verification and validation.

S.N.	Verification	Validation
1	Verification addresses the concern: "Are you building it right?"	Validation addresses the concern: "Are you building the right thing?"
2	Ensures that the software system meets all the functionality.	Ensures that the functionalities meet the intended behavior.
3	Verification takes place first and includes the checking for documentation, code, etc.	Validation occurs after verification and mainly involves the checking of the overall product.
4	Done by developers.	Done by testers.

5	It has static activities, as it includes collecting reviews, walkthroughs, and inspections to verify a software.	It has dynamic activities, as it includes executing the software against the requirements.
6	It is an objective process and no subjective decision should be needed to verify a software.	It is a subjective process and involves subjective decisions on how well a software works.

Software Testing - Myths

Given below are some of the most common myths about software testing.

Myth 1: Testing is Too Expensive

Reality: There is a saying, pay less for testing during software development or pay more for maintenance or correction later. Early testing saves both time and cost in many aspects, however reducing the cost without testing may result in improper design of a software application rendering the product useless.

Myth 2: Testing is Time-Consuming

Reality: During the SDLC phases, testing is never a time-consuming process. However, diagnosing and fixing the errors identified during proper testing is a time-consuming but productive activity.

Myth 3: Only Fully Developed Products are Tested

Reality: No doubt, testing depends on the source code but reviewing requirements and developing test cases is independent from the developed code. However iterative or incremental approach as a development life cycle model may reduce the dependency of testing on the fully developed software.

Myth 4: Complete Testing is Possible

Reality: It becomes an issue when a client or tester thinks that complete testing is possible. It is possible that all paths have been tested by the team but occurrence of complete testing is never

possible. There might be some scenarios that are never executed by the test team or the client during the software development life cycle and may be executed once the project has been deployed.

Myth 5: A Tested Software is Bug-Free

Reality: This is a very common myth that the clients, project managers, and the management team believes in. No one can claim with absolute certainty that a software application is 100% bug-free even if a tester with superb testing skills has tested the application.

Myth 6: Missed Defects are due to Testers

Reality: It is not a correct approach to blame testers for bugs that remain in the application even after testing has been performed. This myth relates to Time, Cost, and Requirements changing Constraints. However, the test strategy may also result in bugs being missed by the testing team.

Myth 7: Testers are Responsible for Quality of Product

Reality: It is a very common misinterpretation that only testers or the testing team should be responsible for product quality. Testers' responsibilities include the identification of bugs to the stakeholders and then it is their decision whether they will fix the bug or release the software. Releasing the software at the time puts more pressure on the testers, as they will be blamed for any error.

Myth 8: Test Automation should be used wherever possible to Reduce Time

Reality: Yes, it is true that Test Automation reduces the testing time, but it is not possible to start test automation at any time during software development. Test automaton should be started when the software has been manually tested and is stable to some extent. Moreover, test automation can never be used if requirements keep changing.

Myth 9: Anyone can Test a Software Application

Reality: People outside the IT industry think and even believe that anyone can test a software and testing is not a creative job. However, testers know very well that this is a myth. Thinking alternative scenarios, try to crash a software with the intent to explore potential bugs is not possible for the person who developed it.

Myth 10: A Tester's only Task is to Find Bugs

Reality: Finding bugs in a software is the task of the testers, but at the same time, they are domain experts of the particular software. Developers are only responsible for the specific component or area that is assigned to them but testers understand the overall workings of the software, what the dependencies are, and the impacts of one module on another module.

Software Testing - QA, QC & Testing

Testing, Quality Assurance, and Quality Control

Most people get confused when it comes to pin down the differences among Quality Assurance, Quality Control, and Testing. Although they are interrelated and to some extent, they can be considered as same activities, but there exist distinguishing points that set them apart. The following table lists the points that differentiate QA, QC, and Testing.

Quality Assurance	Quality Control	Testing
QA includes activities that ensure the implementation of processes, procedures and standards in context to verification of developed software and intended requirements.	It includes activities that ensure the verification of a developed software with respect to documented (or not in some cases) requirements.	It includes activities that ensure the identification of bugs/error/defects in a software.
Focuses on processes and procedures rather than conducting actual testing on the system.	Focuses on actual testing by executing the software with an aim to identify bug/defect through implementation of	Focuses on actual testing.

	procedures and process.	
Process-oriented activities.	Product-oriented activities.	Product-oriented activities.
Preventive activities.	It is a corrective process.	It is a preventive process.
It is a subset of Software Test Life Cycle (STLC).	QC can be considered as the subset of Quality Assurance.	Testing is the subset of Quality Control.

Testing and Debugging

Testing : It involves identifying bug/error/defect in a software without correcting it. Normally professionals with a quality assurance background are involved in bugs identification. Testing is performed in the testing phase.

Debugging : It involves identifying, isolating, and fixing the problems/bugs. Developers who code the software conduct debugging upon encountering an error in the code. Debugging is a part of White Box Testing or Unit Testing. Debugging can be performed in the development phase while conducting Unit Testing or in phases while fixing the reported bugs.

Software Testing - Types of Testing

This section describes the different types of testing that may be used to test a software during SDLC.

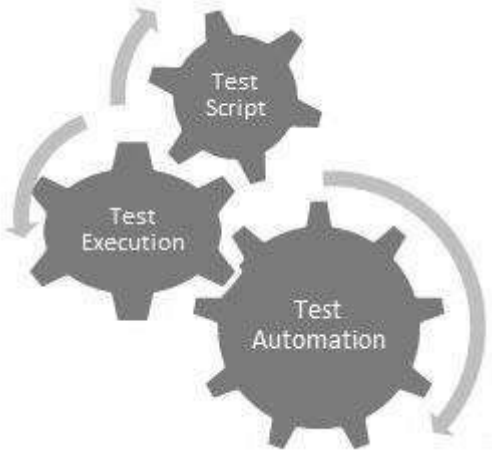
Manual Testing

Manual testing includes testing a software manually, i.e., without using any automated tool or any script. In this type, the tester takes over the role of an end-user and tests the software to identify any unexpected behavior or bug. There are different stages for manual testing such as unit testing, integration testing, system testing, and user acceptance testing.

Testers use test plans, test cases, or test scenarios to test a software to ensure the completeness of testing. Manual testing also includes exploratory testing, as testers explore the software to identify errors in it.

Automation Testing

Automation testing, which is also known as Test Automation, is when the tester writes scripts and uses another software to test the product. This process involves automation of a manual process. Automation Testing is used to re-run the test scenarios that were performed manually, quickly, and repeatedly.



Apart from regression testing, automation testing is also used to test the application from load, performance, and stress point of view. It increases the test coverage, improves accuracy, and saves time and money in comparison to manual testing.

Software Testing - Levels

There are different levels during the process of testing. In this chapter, a brief description is provided about these levels.

Levels of testing include different methodologies that can be used while conducting software testing. The main levels of software testing are:

- Functional Testing
- Non-functional Testing

Functional Testing

This is a type of black-box testing that is based on the specifications of the software that is to be tested. The application is tested by providing input and then the results are examined that need to

conform to the functionality it was intended for. Functional testing of a software is conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements.

There are five steps that are involved while testing an application for functionality.

Steps	Description
I	The determination of the functionality that the intended application is meant to perform.
II	The creation of test data based on the specifications of the application.
III	The output based on the test data and the specifications of the application.
IV	The writing of test scenarios and the execution of test cases.
V	The comparison of actual and expected results based on the executed test cases.

An effective testing practice will see the above steps applied to the testing policies of every organization and hence it will make sure that the organization maintains the strictest of standards when it comes to software quality.

Unit Testing

This type of testing is performed by developers before the setup is handed over to the testing team to formally execute the test cases. Unit testing is performed by the respective developers on the individual units of source code assigned areas. The developers use test data that is different from the test data of the quality assurance team.

The goal of unit testing is to isolate each part of the program and show that individual parts are correct in terms of requirements and functionality.

Non-Functional Testing

This section is based upon testing an application from its non-functional attributes. Non-functional testing involves testing a software from the requirements which are nonfunctional in nature but important such as performance, security, user interface, etc.

Some of the important and commonly used non-functional testing types are discussed below.

Performance Testing

It is mostly used to identify any bottlenecks or performance issues rather than finding bugs in a software. There are different causes that contribute in lowering the performance of a software:

- Network delay
- Client-side processing
- Database transaction processing
- Load balancing between servers
- Data rendering

Performance testing is considered as one of the important and mandatory testing type in terms of the following aspects:

- Speed (i.e. Response Time, data rendering and accessing)
- Capacity
- Stability
- Scalability

Usability Testing

Usability testing is a black-box technique and is used to identify any error(s) and improvements in the software by observing the users through their usage and operation.

- According to Nielsen, usability can be defined in terms of five factors, i.e. Efficiency of use, learn-ability, memory-ability, errors/safety, and satisfaction. According to him, the usability of a product will be good and the system is usable if it possesses the above factors.
- Nigel Bevan and Macleod considered that usability is the quality requirement that can be measured as the outcome of interactions with a computer system. This requirement can be

fulfilled and the end-user will be satisfied if the intended goals are achieved effectively with the use of proper resources.

- Molich in 2000 stated that a user-friendly system should fulfill the following five goals, i.e., easy to Learn, easy to remember, efficient to use, satisfactory to use, and easy to understand.

In addition to the different definitions of usability, there are some standards and quality models and methods that define usability in the form of attributes and sub-attributes such as ISO-9126, ISO-9241-11, ISO-13407, and IEEE std.610.12, etc.

UI vs Usability Testing

UI testing involves testing the Graphical User Interface of the Software. UI testing ensures that the GUI functions according to the requirements and tested in terms of color, alignment, size, and other properties. On the other hand, usability testing ensures a good and user-friendly GUI that can be easily handled. UI testing can be considered as a sub-part of usability testing.

Security Testing

Security testing involves testing a software in order to identify any flaws and gaps from security and vulnerability point of view. Listed below are the main aspects that security testing should ensure:

- Confidentiality
- Integrity
- Authentication
- Availability
- Authorization
- Non-repudiation
- Software is secure against known and unknown vulnerabilities
- Software data is secure
- Software is according to all security regulations
- Input checking and validation

- SQL insertion attacks
- Injection flaws
- Session management issues
- Cross-site scripting attacks
- Buffer overflows vulnerabilities
- Directory traversal attacks

Portability Testing

Portability testing includes testing a software with the aim to ensure its reusability and that it can be moved from another software as well. Following are the strategies that can be used for portability testing:

- Transferring an installed software from one computer to another.
- Building executable (.exe) to run the software on different platforms.

Portability testing can be considered as one of the sub-parts of system testing, as this testing type includes overall testing of a software with respect to its usage over different environments. Computer hardware, operating systems, and browsers are the major focus of portability testing. Some of the pre-conditions for portability testing are as follows:

- Software should be designed and coded, keeping in mind the portability requirements.
- Unit testing has been performed on the associated components.
- Integration testing has been performed.
- Test environment has been established.

Software Testing - Documentation

Testing documentation involves the documentation of artifacts that should be developed before or during the testing of Software.

Documentation for software testing helps in estimating the testing effort required, test coverage, requirement tracking/tracing, etc. This section describes some of the commonly used documented artifacts related to software testing such as:

- Test Plan
- Test Scenario
- Test Case
- Traceability Matrix

Test Plan

A test plan outlines the strategy that will be used to test an application, the resources that will be used, the test environment in which testing will be performed, and the limitations of the testing and the schedule of testing activities. Typically the Quality Assurance Team Lead will be responsible for writing a Test Plan.

A test plan includes the following:

- Introduction to the Test Plan document
- Assumptions while testing the application
- List of test cases included in testing the application
- List of features to be tested
- What sort of approach to use while testing the software
- List of deliverables that need to be tested
- The resources allocated for testing the application
- Any risks involved during the testing process
- A schedule of tasks and milestones to be achieved

Test Scenario

It is a one-line statement that notifies what area in the application will be tested. Test scenarios are used to ensure that all process flows are tested from end to end. A particular area of an application can have as little as one test scenario to a few hundred scenarios depending on the magnitude and complexity of the application.

The terms 'test scenario' and 'test cases' are used interchangeably, however a test scenario has several steps, whereas a test case has a single step. Viewed from this perspective, test scenarios are test cases, but they include several test cases and the sequence that they should be executed. Apart from this, each test is dependent on the output from the previous test.

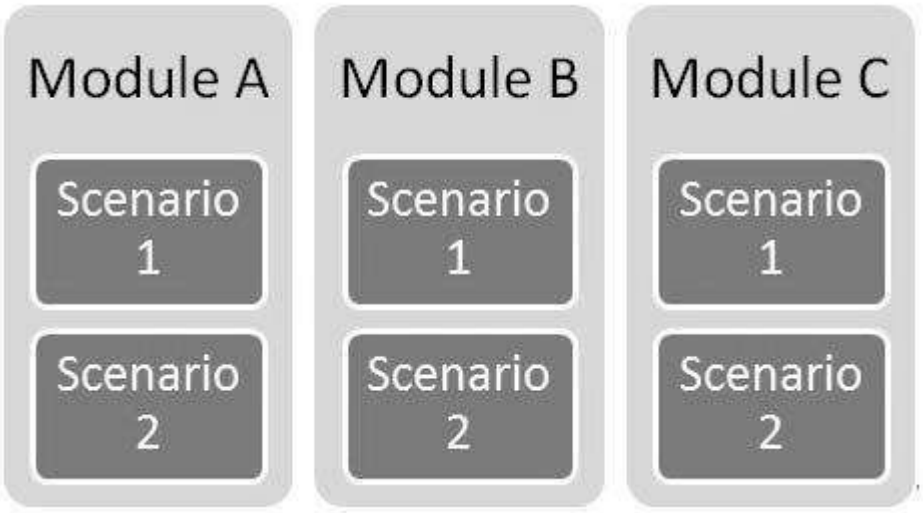


Figure No. 1

Test Case

Test cases involve a set of steps, conditions, and inputs that can be used while performing testing tasks. The main intent of this activity is to ensure whether a software passes or fails in terms of its functionality and other aspects. There are many types of test cases such as functional, negative, error, logical test cases, physical test cases, UI test cases, etc.

Furthermore, test cases are written to keep track of the testing coverage of a software. Generally, there are no formal templates that can be used during test case writing. However, the following components are always available and included in every test case:

- Test case ID
- Product module
- Product version

- Revision history
- Purpose
- Assumptions
- Pre-conditions
- Steps
- Expected outcome
- Actual outcome
- Post-conditions

Many test cases can be derived from a single test scenario. In addition, sometimes multiple test cases are written for a single software which are collectively known as test suites.

Software Testing - Estimation Techniques

Estimating the efforts required for testing is one of the major and important tasks in SDLC. Correct estimation helps in testing the software with maximum coverage. This section describes some of the techniques that can be useful in estimating the efforts required for testing.

Functional Point Analysis

This method is based on the analysis of functional user requirements of the software with the following categories:

- Outputs
- Inquiries
- Inputs
- Internal files
- External files

Test Point Analysis

This estimation process is used for function point analysis for black-box or acceptance testing. The main elements of this method are: Size, Productivity, Strategy, Interfacing, Complexity, and Uniformity.

Miscellaneous

You can use other popular estimation techniques such as:

- Delphi Technique
- Analogy Based Estimation
- Test Case Enumeration Based Estimation
- Task (Activity) based Estimation
- IFPUG method

WHY DO SOFTWARE TESTING?

> "A clever person solves a problem. A wise person avoids it."
> *Albert Einstein*

Why test software? "To find the bugs!" Is the instinctive response and many people, developers and programmers included, think that that's what debugging during development and code reviews is for, so formal testing is redundant at best. But a "bug" is really a problem in the code; software testing is focused on finding defects in the final product. Here are some important defects that better testing would have found.

UNIT TESTING:

A series of stand-alone tests are conducted during Unit Testing. Each test examines an individual component that is new or has been modified. A unit test is also called a module test because it tests the individual units of code that comprise the application.

Each test validates a single module that, based on the technical design documents, was built to perform a certain task with the expectation that it will behave in a specific way or produce specific results. Unit tests focus on functionality and reliability, and the entry and exit criteria can be the same for each module or specific to a particular module. Unit testing is done in a test environment prior to system integration. If a defect is discovered during a unit test, the severity of the defect will dictate whether or not it will be fixed before the module is approved.

INTEGRATION TESTING:

Integration testing examines all the components and modules that are new, changed, affected by a change, or needed to form a complete system. Where system testing tries to minimize outside factors, integration testing requires involvement of other systems and interfaces with other applications, including those owned by an outside vendor, external partners, or the customer. For example, integration testing for a new web interface that collects user input for addition to a database must include the database's ETL application even if the database is hosted by a vendor the complete system must be tested end-to-end. In this example, integration testing doesn't stop with the database load; test reads must verify that it was correctly loaded.

Integration testing also differs from system testing in that when a defect is discovered, not all previously executed tests have to be rerun after the repair is made. Only those tests with a connection to the defect must be rerun, but retesting must start at the point of repair if it is before the point of failure. For example, the retest of a failed FTP process may use an existing data file instead of recreating it if up to that point everything else was OK.

Ch# 6

Screenshots

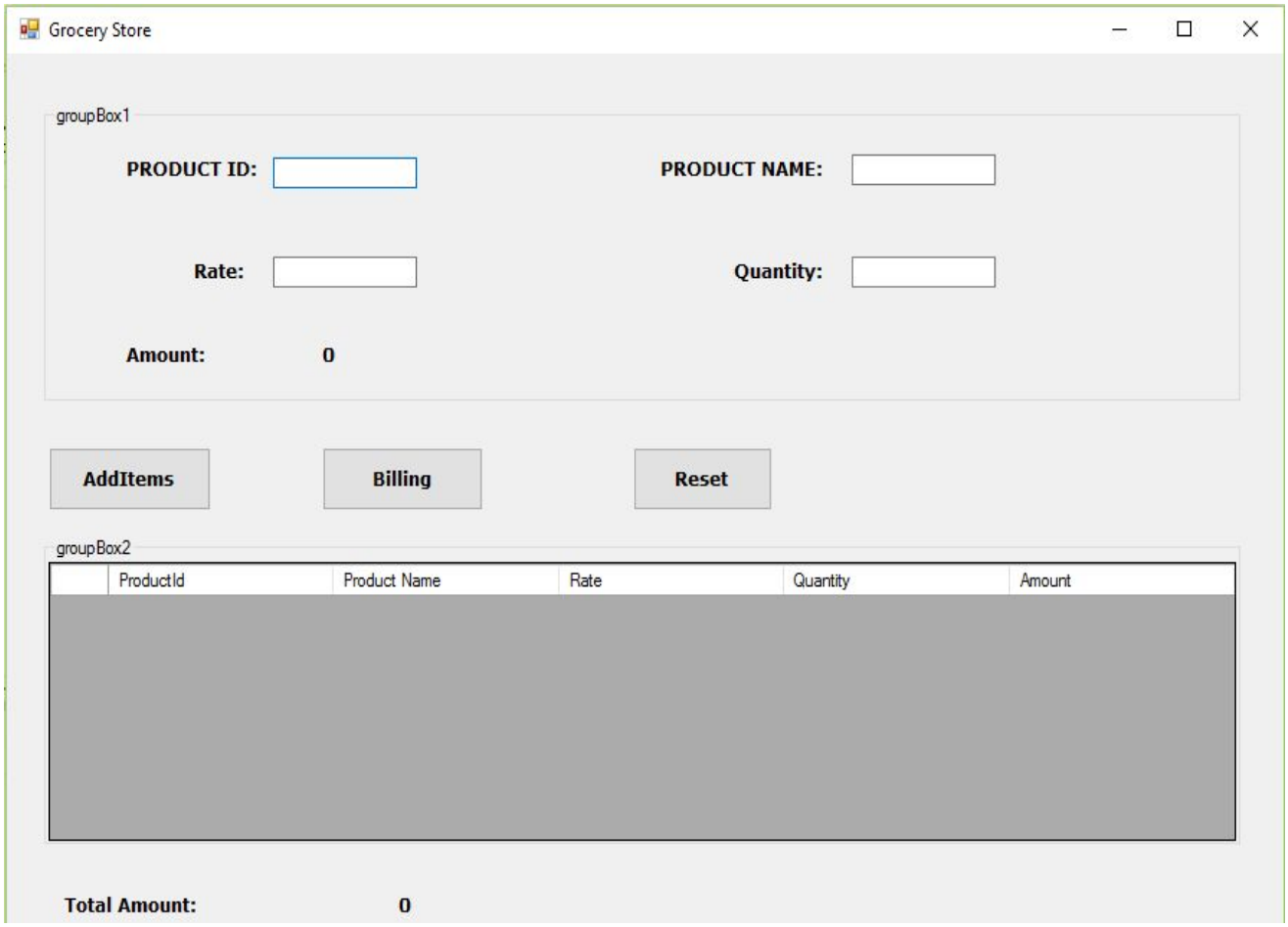

Grocery Store
groupBox1
PRODUCT ID:
PRODUCT NAME:
Rate:
Quantity:
Amount:
0
AddItems
Billing
Reset
groupBox2
ProductId
Product Name
Rate
Quantity
Amount
Total Amount:
0

Figure No. 1

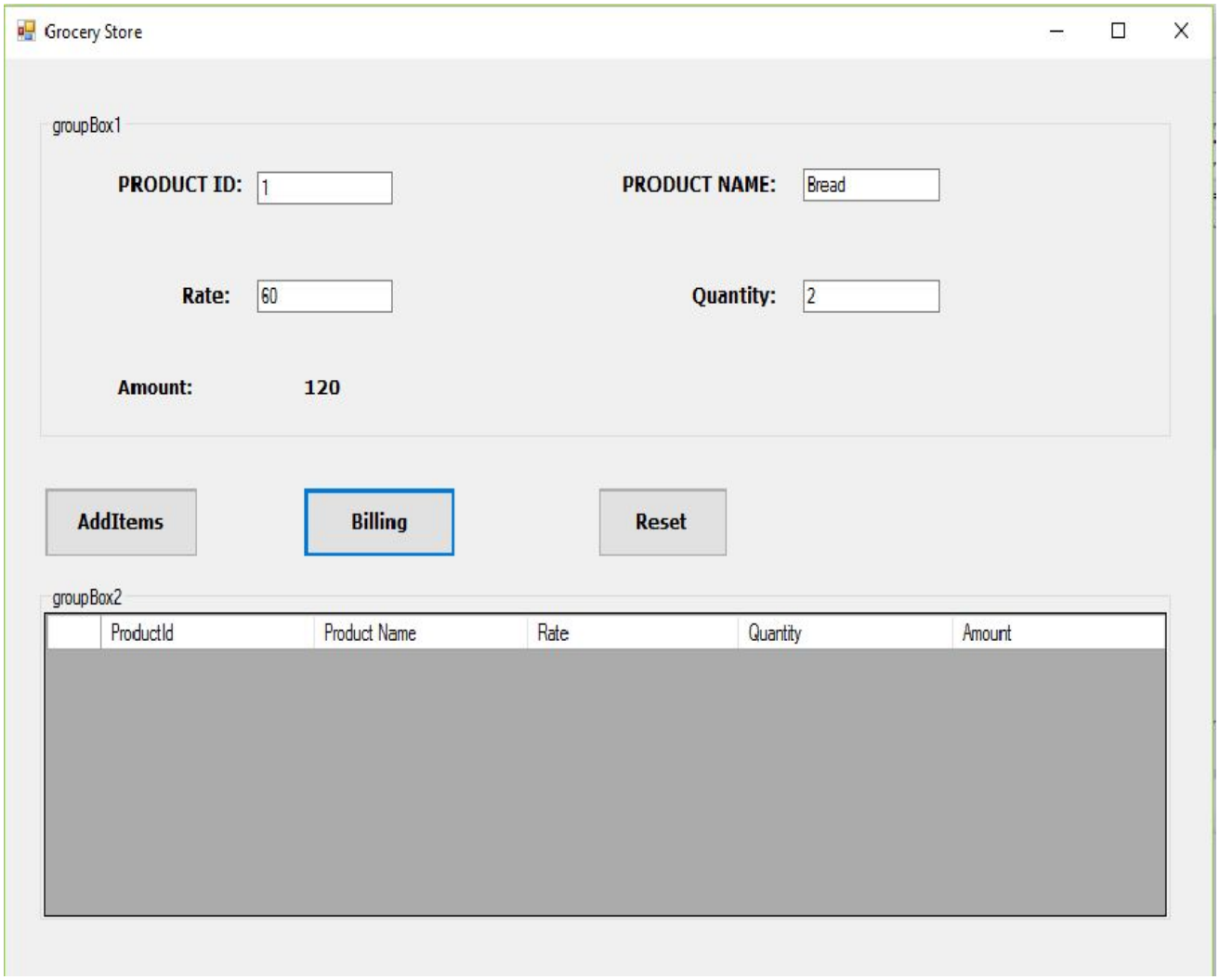

Grocery Store
groupBox1
PRODUCT ID:
1
PRODUCT NAME:
Bread
Rate:
60
Quantity:
2
Amount:
120
AddItems
Billing
Reset
groupBox2
ProductId
Product Name
Rate
Quantity
Amount

Figure No. 2

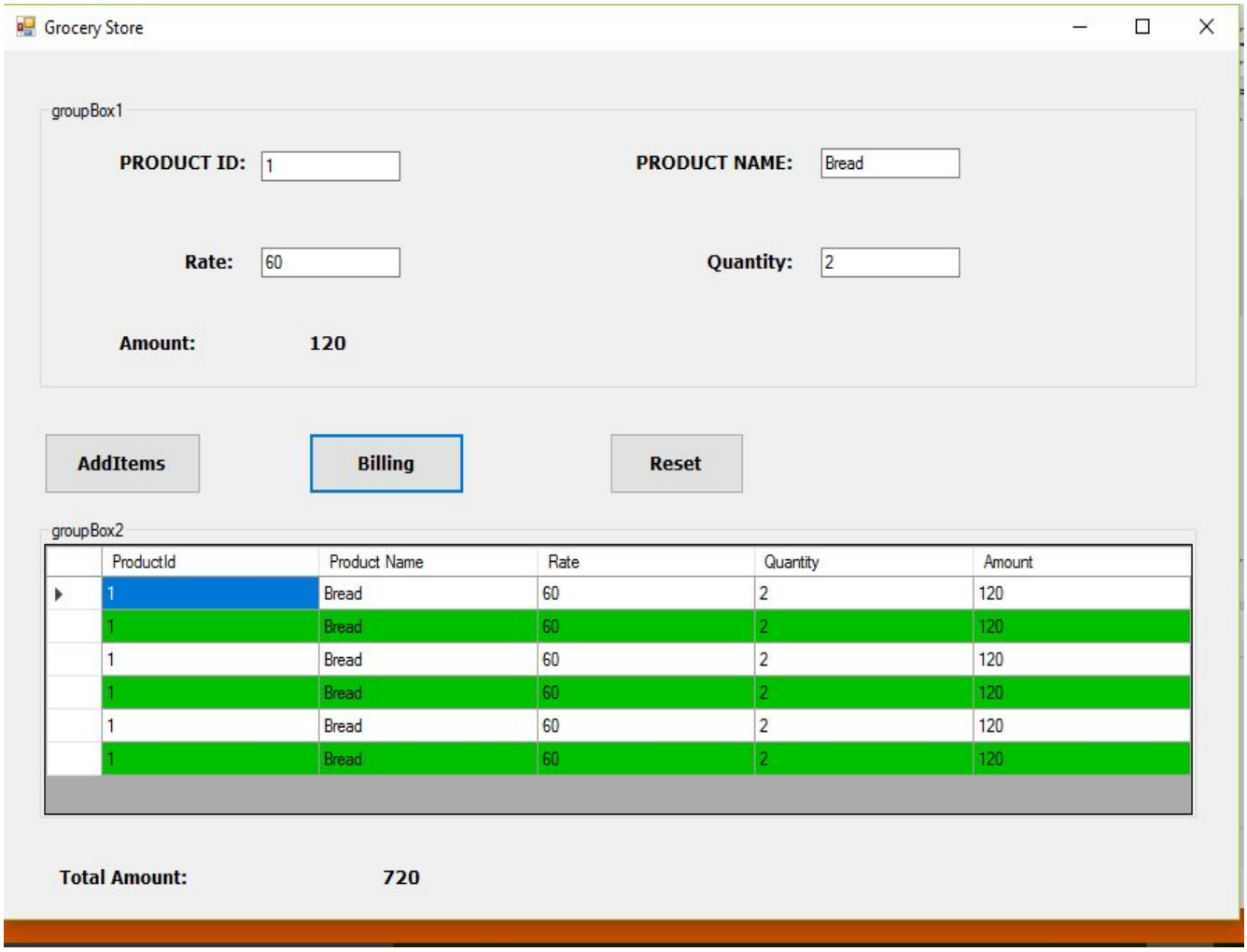

Grocery Store
groupBox1
PRODUCT ID:
1
PRODUCT NAME:
Bread
Rate:
60
Quantity:
2
Amount:
120
AddItems
Billing
Reset
groupBox2
ProductId
Product Name
Rate
Quantity
Amount
1 Bread 60 2 120
1 Bread 60 2 120
1 Bread 60 2 120
1 Bread 60 2 120
1 Bread 60 2 120
1 Bread 60 2 120
Total Amount:
720

Figure No. 3

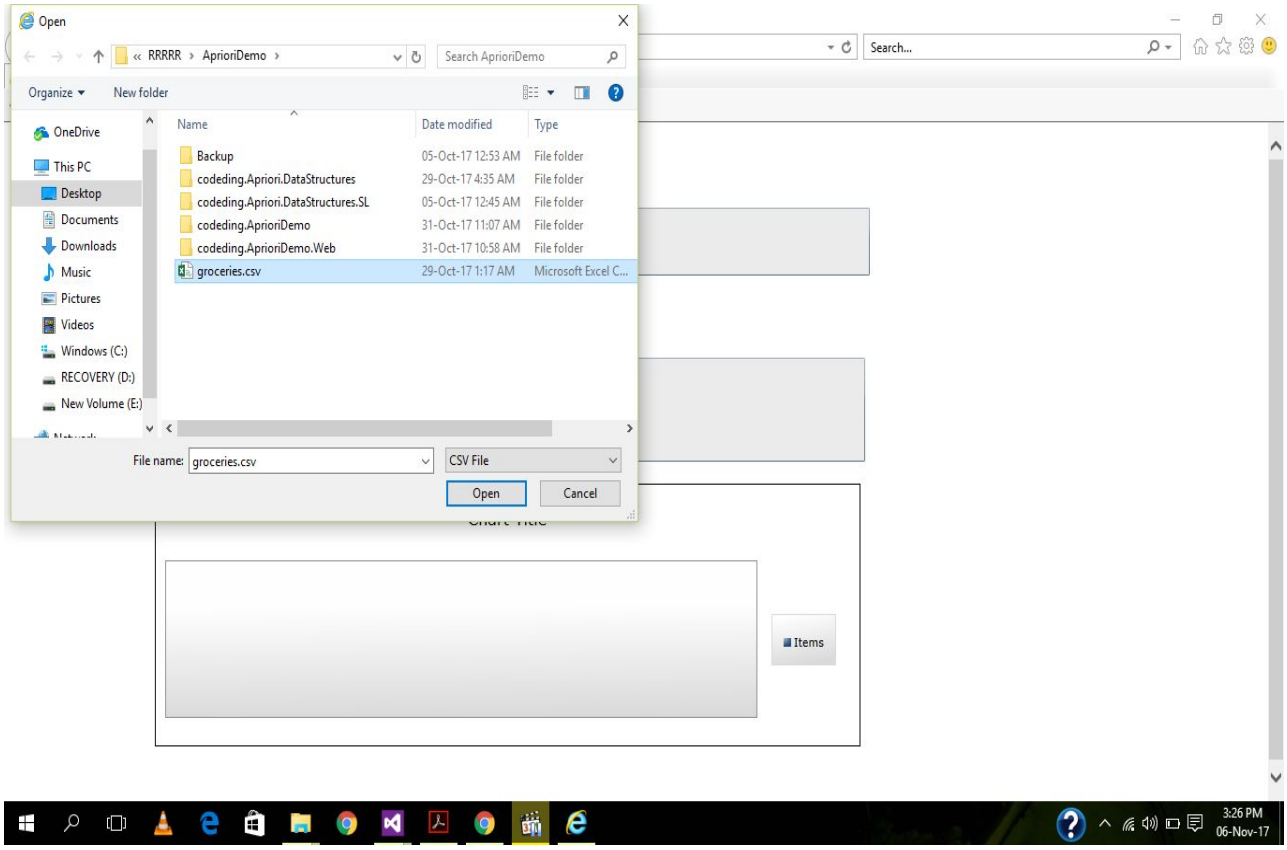

Open
RRRRR › AprioriDemo ›
Search AprioriDemo
Organize
New folder
OneDrive
This PC
Desktop
Documents
Downloads
Music
Pictures
Videos
Windows (C:)
RECOVERY (D:)
New Volume (E:)
Name
Date modified
Type
Backup
05-Oct-17 12:53 AM
File folder
codeding.Apriori.DataStructures
29-Oct-17 4:35 AM
File folder
codeding.Apriori.DataStructures.SL
05-Oct-17 12:45 AM
File folder
codeding.AprioriDemo
31-Oct-17 11:07 AM
File folder
codeding.AprioriDemo.Web
31-Oct-17 10:58 AM
File folder
groceries.csv
29-Oct-17 1:17 AM
Microsoft Excel C...
File name:
groceries.csv
CSV File
Open
Cancel
Search...
Items
3:26 PM
06-Nov-17

Figure No. 4

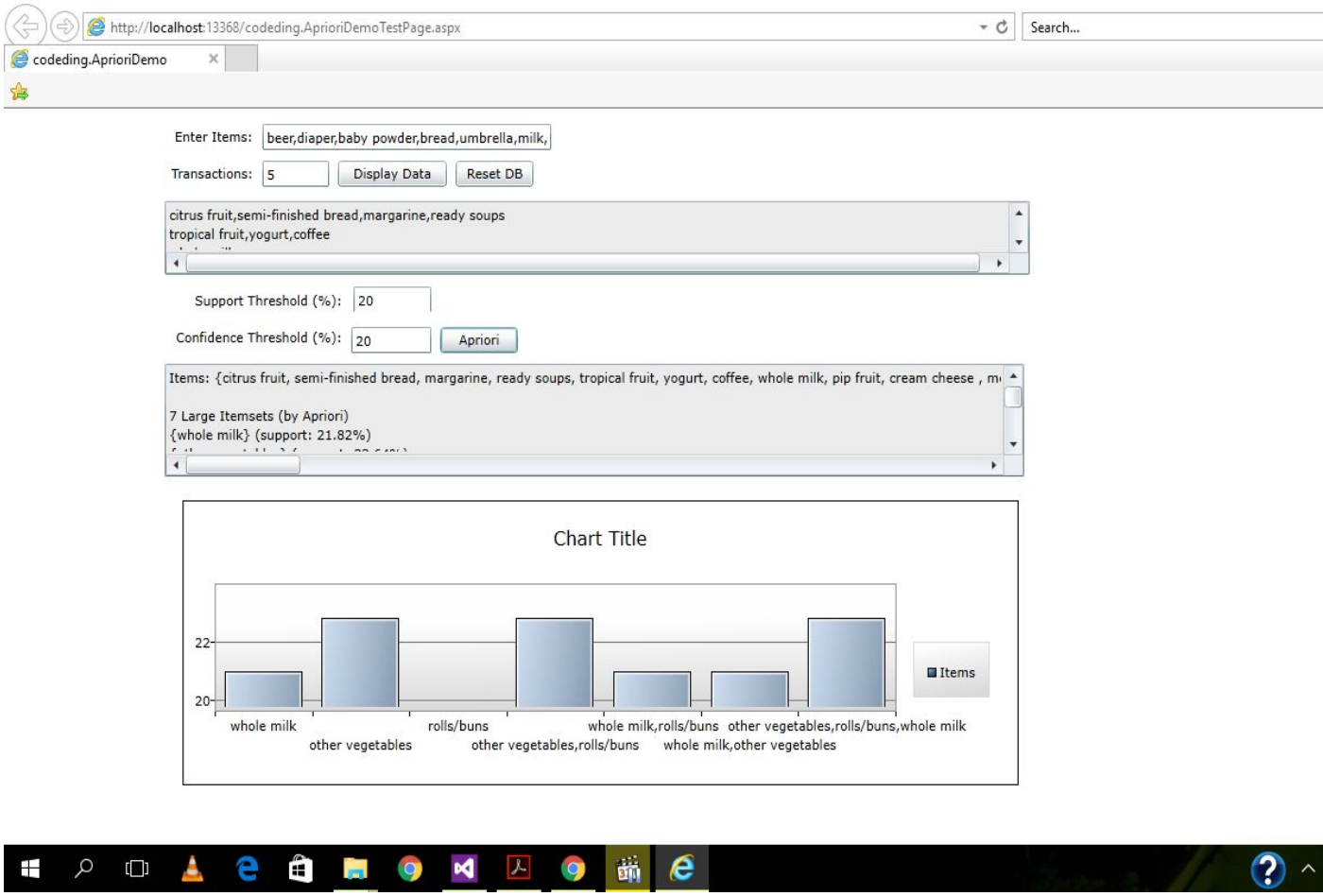

http://localhost:13368/codeding.AprioriDemoTestPage.aspx
Search...
codeding.AprioriDemo
Enter Items:
beer,diaper,baby powder,bread,umbrella,milk,
Transactions:
5
Display Data
Reset DB
citrus fruit,semi-finished bread,margarine,ready soups
tropical fruit,yogurt,coffee
Support Threshold (%):
20
Confidence Threshold (%):
20
Apriori
Items: {citrus fruit, semi-finished bread, margarine, ready soups, tropical fruit, yogurt, coffee, whole milk, pip fruit, cream cheese ,
7 Large Itemsets (by Apriori)
{whole milk} (support: 21.82%)
Chart Title
22
20
Items
whole milk
other vegetables
rolls/buns
other vegetables,rolls/buns
whole milk,rolls/buns
whole milk,other vegetables
other vegetables,rolls/buns,whole milk

Figure No. 5