INVENTORY MANAGEMENT SYSTEM

Submitted to:

Dr. Shelly Sachdeva

Head of Department

Computer Science

NIT Delhi

Submitted by:

Hitendra (171210028)

Ankit (171210009)

Irshad (171210029)

Akhil (171210005)

CONTENTS

INTRODUCTION

In this world, at this time, we have a ubiquitous and essential entity called database management system. It is a software used to manage databases. Database management has evolved from a specialized computer application to a central component of a modern computing environment, and, as a result, knowledge about database systems has become an essential part of an education in computer science.

The project INVENTORY MANAGEMENT SYSTEM is an example of working database system where we manage inventory and analyze the day-to-day records of a dairy.

LIST OF ABBREVIATIONS

- **IMS :** Inventory Management System
- **DBMS :** Database Management System
- **DDL :** Data Definition Language
- **DML :** Data Manipulation Language
- **SQL :** Structured Query Language
- **SSMS :** SQL Server Management Studio Express
- **ER :** Entity Relationship

DECLARATION

We hereby declare that the work reported in this project work on “Inventory Management System” submitted to National Institute of Technology, Delhi is our original work done under the supervision of Dr. Shelly Sachdeva, Head of Department (Computer Science), National Institute of Technology, Delhi. The material contained in the report has not been submitted to any university or institution for any award of any degree.

ACKNOWLEDGEMENT

This project is prepared in the partial fulfillment of the requirement for the degree of Bachelor's in Technology in Computer Science. The satisfaction and completion of this task would be incomplete without the heartfelt thanks to people whose constant guidance, support and encouragement made this work successful. On doing this undergraduate project we have been fortunate to have help, support and encouragement from many people we would like to acknowledge them for their cooperation.

Our essence spirit of gratitude to National Institute of Technology, Delhi for designing such a worthy syllabus and making us do this project. Our next essence of gratitude goes to Dr. Shelly Sachdeva, Head of Department (Computer Science) without whose help our project would have never been possible to bring into existence. We would also like to thank all the batch mates who, without any hesitation, came to our aid whenever needed. Last but not the least we want to thank every direct and indirect hands that were involved in completion of this project.

This project has been a learning curve for all of us which strengthened the concepts of database functionality, and it has been a wonderful experience where we have learnt and experienced many beneficial things.

With Regards

Hitendra (171210028)

Ankit (171210009)

Irshad (171210029)

Akhil (171210005)

THEORY

Database Theory:

A database is a collection of information that is organizes so that it can easily be accessed, managed and updated. In one view, database can be classified according to types of content: bibliography, full-text, numeric, and image. In computing, database is sometime classified according to their organizational approach. A distributed database is one that can be dispersed or replicated among different points in a network.

Relational Database:

IMS has the relational database model. A relational database is a digital database whose organization is based on the relational model of data. This model organizes data into one or more tables of rows and columns. These tables here have the relation.

The relation is maintained by the unique key defined in each row. The key can be primary and foreign depending on their nature of connection. The standard user and application program interface to a relational database is the structured query language (SQL). SQL statement are used both for interactive queries for information from relational database and for gathering data for reports.

Primary Key:

The primary key of a relational table uniquely identifies each record in the table. It can either be a normal attribute that is guaranteed to be unique or it can be generated by the DBMS. A primary key's main features are:

- It must contain a unique value for each row of data.
- It cannot contain null value.

Foreign Key:

A foreign key is a column or group of column in a relational database table that provides a link between data in two tables. In foreign key reference, a link is created between two tables when the column or columns that hold the primary key value for one table are referenced by the column or column in another table thereby establishing a link between them. Creating a foreign key manually includes the following advantages:

- Changes to primary key constraints are checked with foreign key constraints in relation table.
- An index enables the Database Engine to quickly find related data in the foreign key tables.

Structured Query Language (SQL):

The structured Query language (SQL) is the set of instructions used to interact with a relational database. In fact, SQL is the only language the most database actually understands. Whenever you interact with such a database, the software translates your commands into SQL statement that the database knows how to interpret. SQL has three major Components:

- Data Manipulation Language (DML)
- Data Definition Language (DDL)
- Data Control Language (DCL)

ACID Property:

Every database transaction obeys the following rules:

Atomicity – Either the effects of all or none of its operation remain ("all or nothing" semantics) when a transaction is completed (committed or aborted respectively). In other words, to the outside world a committed transaction appears (by its effects on the database) to be indivisible, atomic, and an aborted transaction does not leave effects on the database at all, as if never existed.

Consistency – every transaction must leave the database in a consistent (correct) state, i.e., maintain the predetermined integrity rules of the database (constraints upon and among the database's objects). A transaction must transform a database from one consistent state to another consistent state (however, it is the responsibility of the transaction's programmer to make sure that the transaction itself is correct, i.e., performs correctly what it intends to perform (from the application's point of view) while the predefined integrity rules are enforced by the DBMS). Thus since a database can be normally changed only by transactions, all the database's states are consistent. An aborted transaction does not change the database state it has started from, as if it never existed (atomicity above).

Isolation – Transactions cannot interfere with each other (as an end result of their executions). Moreover, usually (depending on concurrency control method) the effects of an incomplete transaction are not even visible to another transaction. Providing isolation is the main goal of concurrency control.

Durability – Effects of successful (committed) transactions must persist through crashes (typically) by recording the transaction's effects and its commit event in a non-volatile memory.

FUNCTIONALITIES & REQUIREMENTS

1. **Total Sales** – Date wise, Product wise, Week wise, Month wise, Previous Months Data, Variance – Day wise, Week wise, Month wise --- **All in graphs**
2. **Products**: Milk, Ghee – 2 types, Curd, Nasika, Ark – 3 types, Phenyl, Mustard Oil
3. Customer Orders: In order to see the **seasonal effect in sale** we need to trace out orders from customers for our products, though it can be tracked with sales also.
4. Sales - **Target vs Actual**, **Profit vs Loss** - Monthly
5. Auto **message if loss is more than 5%** in each product and cumulative(annual) sales.
6. **Capital Items Tracing** --- Purchase date and Total amount, Depreciation, etc.
7. **Connectivity with customers** – as soon as we are launching new products or stores.
8. Tracking of **customer feedback and suggestions**.
9. **Which Customers are using our products frequently** – Product wise/ Month Wise (SQL Query)

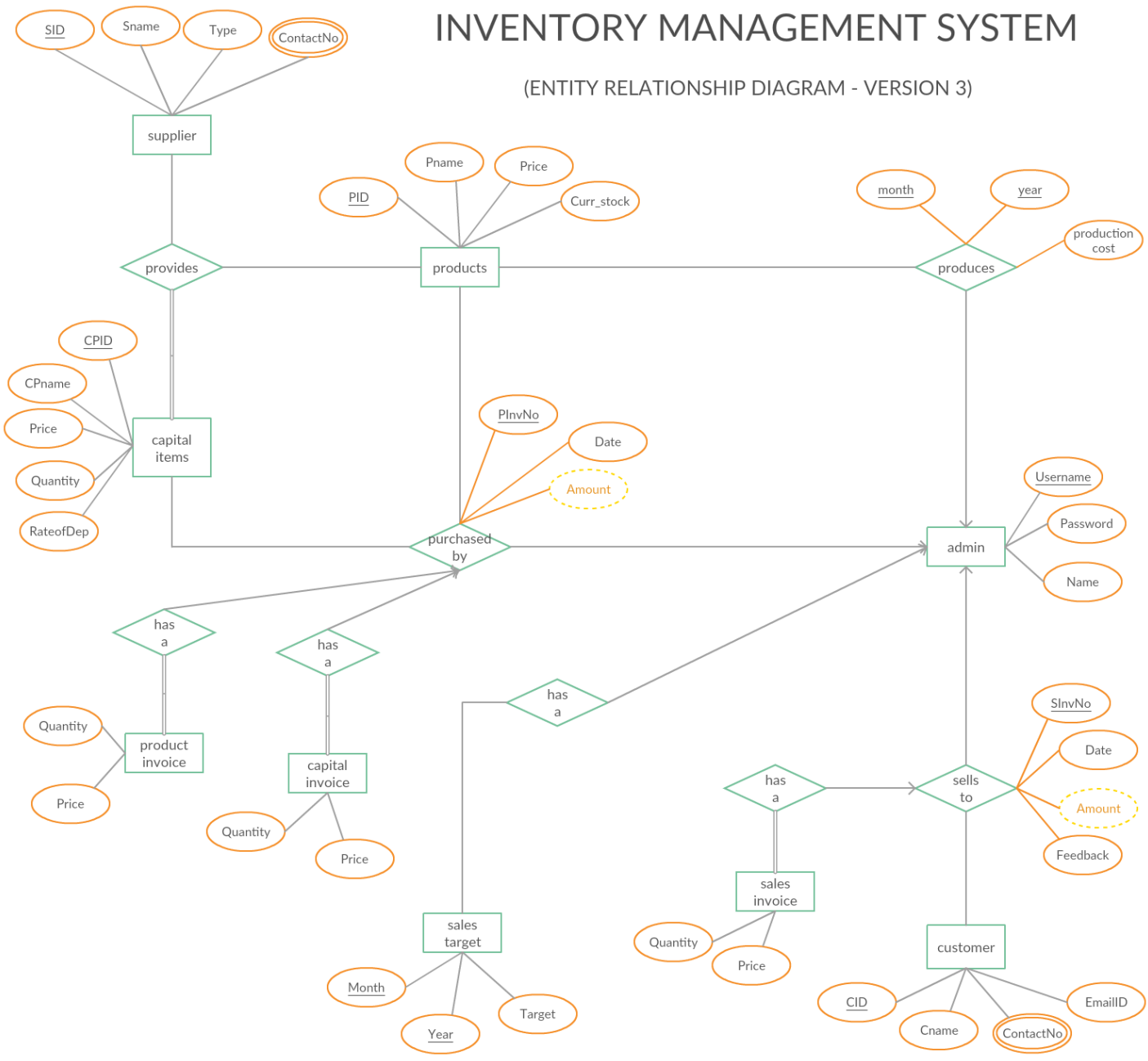
Entity-Relationship Diagram

On the next page, is attached the approved ER diagram of Inventory Management System.

(Please turn over)

INVENTORY MANAGEMENT SYSTEM

(ENTITY RELATIONSHIP DIAGRAM - VERSION 3)



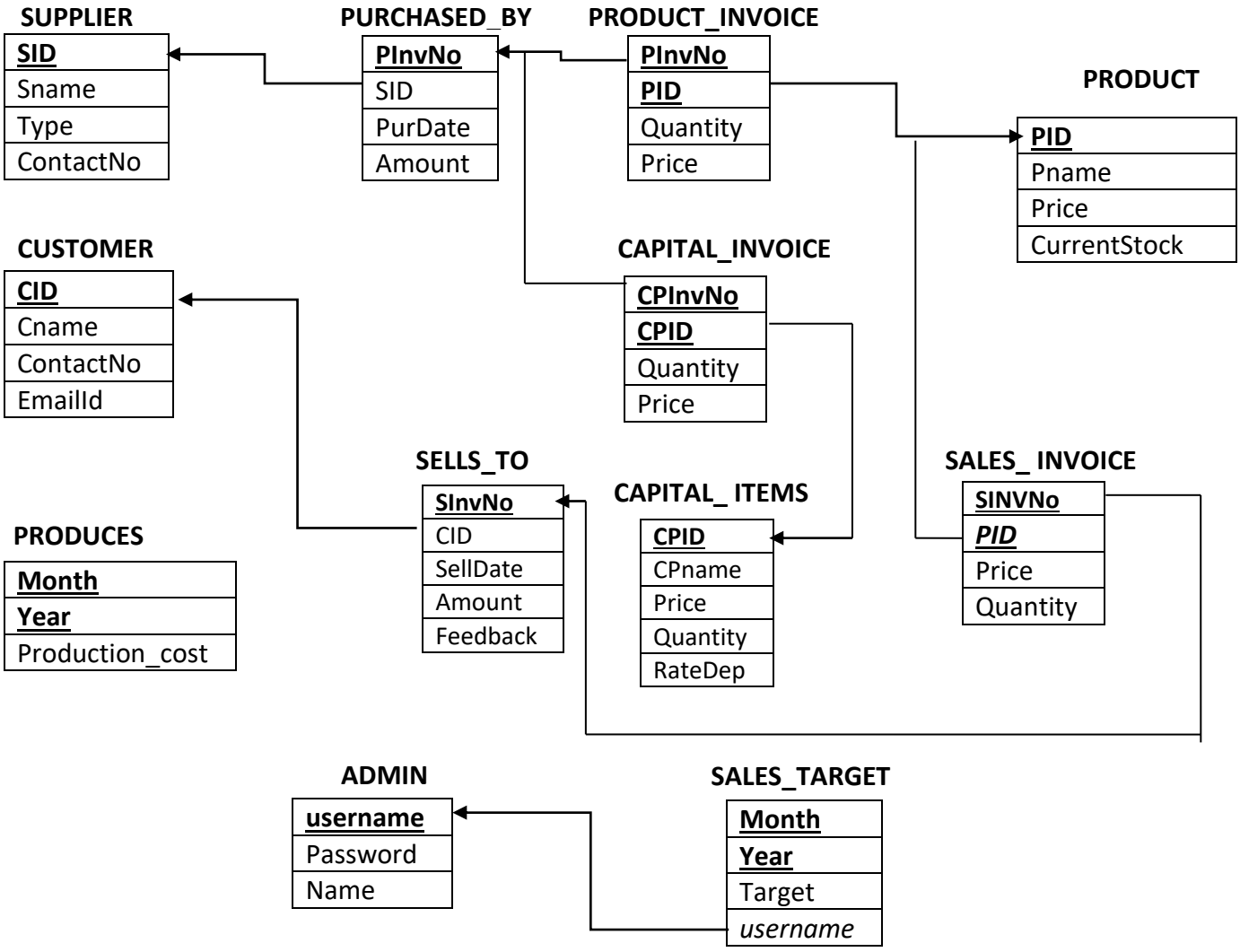
* sellsto (amount) derived from salesinvoice (Price)

* purchasedby (amount) derived from productinvoice (Price) & capitalinvoice (Price)

NORMALIZED DATABASE SCHEMA

- Following is the database schema after passing through the normalization stages:
- 1NF – First Normalized Form, 2NF – Second Normalized Form, 3NF – Third Normalized Form, BCNF – Boyce-Codd Normalized Form

The final schema is:



RELATIONAL ALGEBRA OPERATIONS USED

- **SELECT(σ)**

  -Notation: σ p(r)

  -p is called the selection predicate

  -Defined as: σp(r) = {t | t $\in$ r and p(t)} Where p is a formula in propositional calculus consisting of terms connected by : $\wedge$ (and), $\vee$ (or), $\neg$ (not)

  Each term is one of:

  <attribute> op <attribute> or <constant>

  where *op* is one of: =, $\neq$, >, $\geq$. <. $\leq$

- **PROJECT (π)**

  -Notation: $\prod_{A_1, A_2, \ldots, A_k}(r)$

  where A1, A2 are attribute names and r is a relation name.

  The result is defined as the relation of k columns obtained by erasing the columns that are not listed and duplicate rows removed from result, since relations are sets

➢ CARTESIAN PRODUCT (A X B)

Notation r x s

Defined as: r x s = {t q | t $\in$ r and q $\in$ s}

Assume that attributes of r(R) and s(S) are disjoint. (That is, R $\cap$ S = $\varnothing$).

If attributes of r(R) and s(S) are not disjoint, then renaming must be used.

➢ JOIN ($\bowtie$)

Let r and s be relations on schemas R and S respectively.

Then, r s is a relation on schema R $\cup$ S obtained as follows:

- Consider each pair of tuples tr from r and ts from s.
- If tr and ts have the same value on each of the attributes in R $\cap$ S, add a tuple t to the result, where
  - t has the same value as tr on r
  - t has the same value as ts on s

- **UNION (∪)**

  Notation: $r \cup s$

  Defined as: $r \cup s = \{t \mid t \in r \text{ or } t \in s\}$

  For $r \cup s$ to be valid.

  1. r, s must have the same arity (same number of attributes)
  2. The attribute domains must be compatible (example: 2nd column of r deals with the same type of values as does the 2nd column of s)

- **INTERSECTION (∩)**

  Notation: $r \cap s$

  Defined as :$r \cap s = \{ t \mid t \in r \text{ and } t \in s \}$

  Assume: r, s have the same attributes of r and s are compatible

DDL QUERIES INCORPORATED

- ***CREATE DATABASE***

  Creates a database repository for the project.

```
CREATE DATABASE INVENTORY;
USE INVENTORY;
```

- ***CREATING TABLES***

```
CREATE TABLE supplier (
Sid INT NOT NULL,
Sname VARCHAR(20) NOT NULL,
Type CHAR(1) NOT NULL,
ContactNo NUMBER(10),
PRIMARY KEY(Sid));

CREATE TABLE products (
Pid INT NOT NULL,
Pname VARCHAR(20) NOT NULL,
Price INT NOT NULL,
CurrentStock INT  DEFAULT 0,
PRIMARY KEY(Pid));

CREATE TABLE addmin (
username VARCHAR(20) NOT NULL,
password VARCHAR(20) NOT NULL,
Name VARCHAR(20),
PRIMARY KEY(username));

CREATE TABLE customer (
Cid INT NOT NULL,
Cname VARCHAR(20) NOT NULL,
ContactNo NUMBER(10),
EmailId VARCHAR(20),
PRIMARY KEY(Cid)
);
CREATE TABLE salestarget (
Mnth INT NOT NULL,
Yr INT NOT NULL,
Target INT,
username VARCHAR(20) NOT
NULL,PRIMARY KEY(Mnth, Yr),
CONSTRAINT
salestarget_username_FK FOREIGN
KEY(username) REFERENCES
addmin(username));

CREATE TABLE produces (
Mnth INT NOT NULL,
Yr INT NOT NULL,
Production_cost INT,
PRIMARY KEY(Mnth, Yr));

CREATE TABLE purchasedby (
Sid INT NOT NULL,
PInvNo INT NOT NULL,
PurDate DATE NOT NULL,
Amount INT NOT NULL,
```

```
PRIMARY KEY(PInvNo),
CONSTRAINT purchasedby_Sid_FK
FOREIGN KEY(Sid) REFERENCES
supplier(Sid));


CREATE TABLE capitalitems (
CPid INT NOT NULL,
CPname VARCHAR(20) NOT NULL,
Price INT,
Quantity INT NOT NULL,
RateDep NUMERIC(4,2),
PRIMARY KEY(CPid));


CREATE TABLE sellsto (
Cid INT NOT NULL,
SInvNo INT NOT NULL,
SellDate DATE NOT NULL,
Amount INT NOT NULL,
Feedback VARCHAR(50),
PRIMARY KEY(SInvNo));

CREATE TABLE salesinvoice (
Pid INT NOT NULL,
SInvNo INT NOT NULL,
Quantity INT NOT NULL,
Price INT NOT NULL,

CONSTRAINT
salesinvoice_SInvNo_FK FOREIGN
KEY(SInvNo) REFERENCES
sellsto(SInvNo),
CONSTRAINT salesinvoice_Pid_FK
FOREIGN KEY(Pid) REFERENCES
products(Pid)
);


CREATE TABLE capitalinvoice (
CPid INT NOT NULL,
CPInvNo INT NOT NULL,
Quantity INT NOT NULL,
Price INT NOT NULL,

CONSTRAINT
capitalinvoice_CPid_FK FOREIGN
KEY(CPid) REFERENCES
capitalitems(CPid),
CONSTRAINT
capitalinvoice_CPInvNo_FK
FOREIGN KEY(CPInvNo) REFERENCES
purchasedby(PInvNo)
);


CREATE TABLE productinvoice (
Pid INT NOT NULL,
PInvNo INT NOT NULL,
Quantity INT NOT NULL,
Price INT NOT NULL,

CONSTRAINT productinvoice_Pid_FK
FOREIGN KEY(Pid) REFERENCES
products(Pid),
CONSTRAINT
productinvoice_PInvNo_FK FOREIGN
KEY(PInvNo) REFERENCES
purchasedby(PinvNo)
);
```

DML QUERIES INCORPORATED

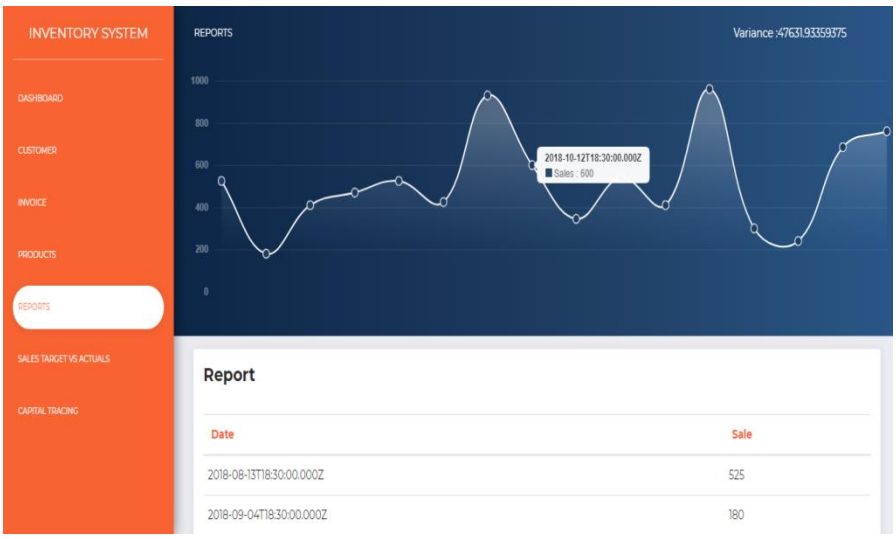
- ***Display the total sales and variance date wise, month wise, week wise, product wise, Season Wise:***

1. **Date wise:**

The user enters the date range – From **DATE1** to **DATE2** and clicks on submit button. The tabular and graphical representation of total sales per day and the variance from **DATE1** to **DATE2** is displayed.

If user enters the range as ’2018-01-01’ to ’2018-12-31’ the query and corresponding output is as:

```
SELECT SellDate , SUM(Amount) AS TotalSale FROM sellsto
WHERE SellDate  BETWEEN '2018-01-01' AND '2018-12-31'
GROUP BY SellDate ORDER BY SellDate;

SELECT variance(TotalSale) AS var FROM
(SELECT SellDate , SUM(Amount) AS TotalSale FROM sellsto
 WHERE SellDate BETWEEN '2018-01-01'AND'2018-12-31' GROUP BY SellDate
) AS saleTable;
```



	SellDate	TotalSale
▶	2018-08-14	525
	2018-09-05	180
	2018-09-19	410
	2018-09-20	470
	2018-10-03	525
	2018-10-04	425
	2018-10-08	930
	2018-10-13	600

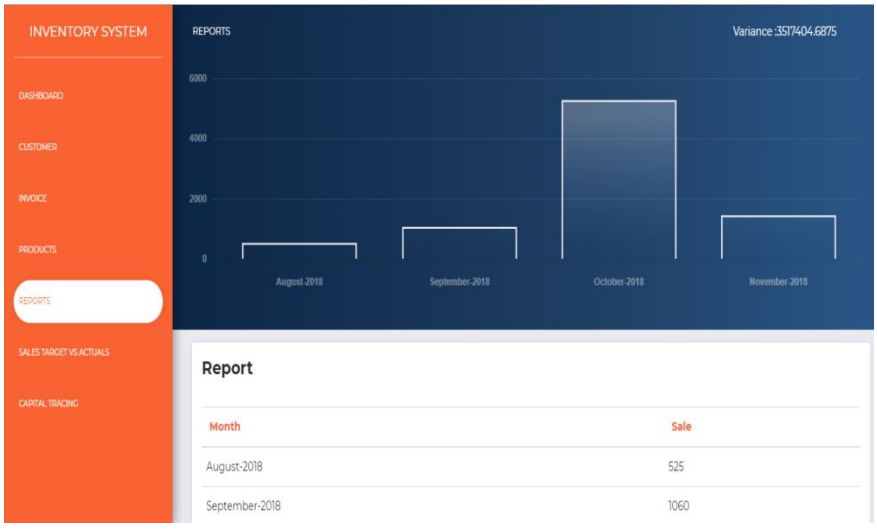
	var
▶	47631.93359375

2. Month wise:

The user enters the range – From month **M1** to **M2** and year **Y1** to **Y2** clicks on submit button. The tabular and graphical representation of total sales within the range and the variance is displayed.

If user enters the range as month 1 to 12 and year 2018 to 2018 the query and corresponding output is as:

```
SELECT CONCAT(MONTHNAME(SellDate),'-',YEAR(SellDate)) AS sell_month ,
SUM(Amount) AS total_sales
FROM sellsto WHERE MONTH(SellDate) BETWEEN 1 AND 12
AND YEAR(SellDate) BETWEEN 2018 AND 2018
GROUP BY sell_month ORDER BY YEAR(SellDate),MONTH(SellDate);

SELECT variance(total_sales) as var
FROM (SELECT CONCAT(MONTHNAME(SellDate),'-',YEAR(SellDate)) AS
sell_month,SUM(Amount) AS total_sales
FROM sellsto WHERE MONTH(SellDate) BETWEEN 1 AND 12
AND YEAR(SellDate) BETWEEN 2018 AND 2018
GROUP BY sell_month
ORDER BY YEAR(SellDate),MONTH(SellDate)) AS saleTable ;
```



	sell_month	total_sales
▶	August-2018	525
	September-2018	1060
	October-2018	5275
	November-2018	1445

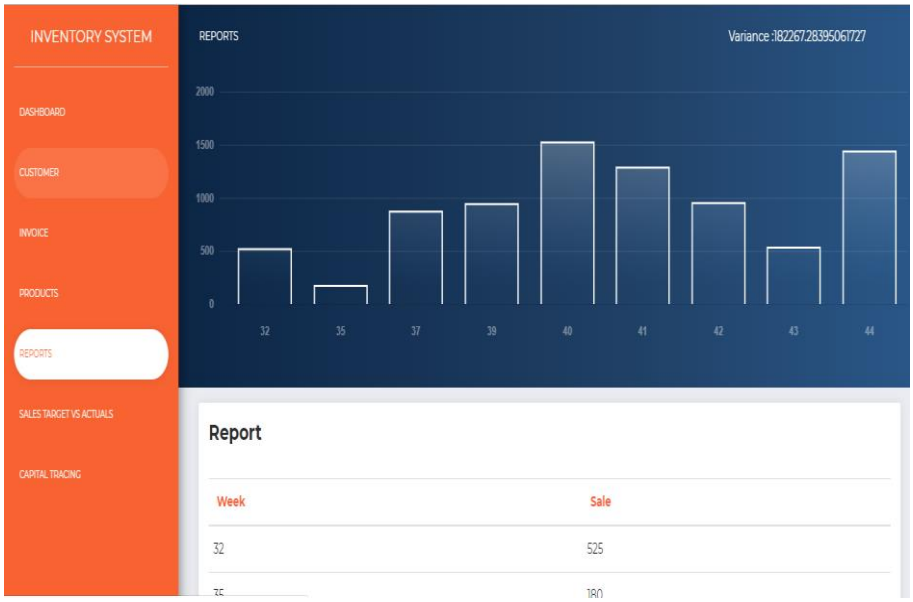
	var
▶	3517404.6875

3. Week wise:

The user enters a year **Y1** and clicks on submit button. The tabular and graphical representation of total sales on each week of that year and the variance is displayed.

If user enters a year 2018, the query and corresponding output is as:

```
SELECT WEEK(SellDate) as week_wise,SUM(Amount) AS total_sales
FROM sellsto WHERE YEAR(SellDate) = 2018
GROUP BY week_wise
ORDER BY week_wise;

SELECT variance(total_sales) AS var
FROM (SELECT WEEK(SellDate) as week_wise,SUM(Amount) AS total_sales
      FROM sellsto WHERE YEAR(SellDate) = 2018
          GROUP BY week_wise
          ORDER BY week_wise
     )AS saleTable;
```



	week_wise	total_sales
▶	32	525
	35	180
	37	880
	39	950
	40	1530
	41	1295
	42	960
	43	540
	44	1445

	var
▶	182267.28395061727

4. Product wise:

The user enters the date range – From **DATE1** to **DATE2** and clicks on submit button. The tabular and graphical representation of total sales per product for the given range and the variance is displayed.

If user enters the range as ’2018-01-01’ to ’2018-12-31’ the query and corresponding output is as:

```
SELECT Pname, SUM(si.Price) as total_sale
FROM sellsto AS s, salesinvoice AS si, products AS p
WHERE s.SInvNo = si.SInvNo AND si.Pid = p.Pid AND
      SellDate  BETWEEN ‘2018-01-01’ AND ‘2018-12-31’
GROUP BY Pname;
```

```
SELECT variance(total_sale) as var FROM (
SELECT Pname, SUM(si.Price) as total_sale
FROM sellsto AS s, salesinvoice AS si, products AS p
WHERE s.SInvNo = si.SInvNo AND si.Pid = p.Pid AND
      SellDate  BETWEEN ‘2018-01-01’ AND ‘2018-12-31’
GROUP BY Pname
     ) as saleTable;
```

	Pname	total_sale
▶	Milk Amul	760
	Ghee Amul	1020
	Ark1	70
	Oil Mustard	2040
	Ghee Ananda	1680
	Curd	180
	Phenyl	500
	Ark3	110
	Nasika	100
	Ark2	50

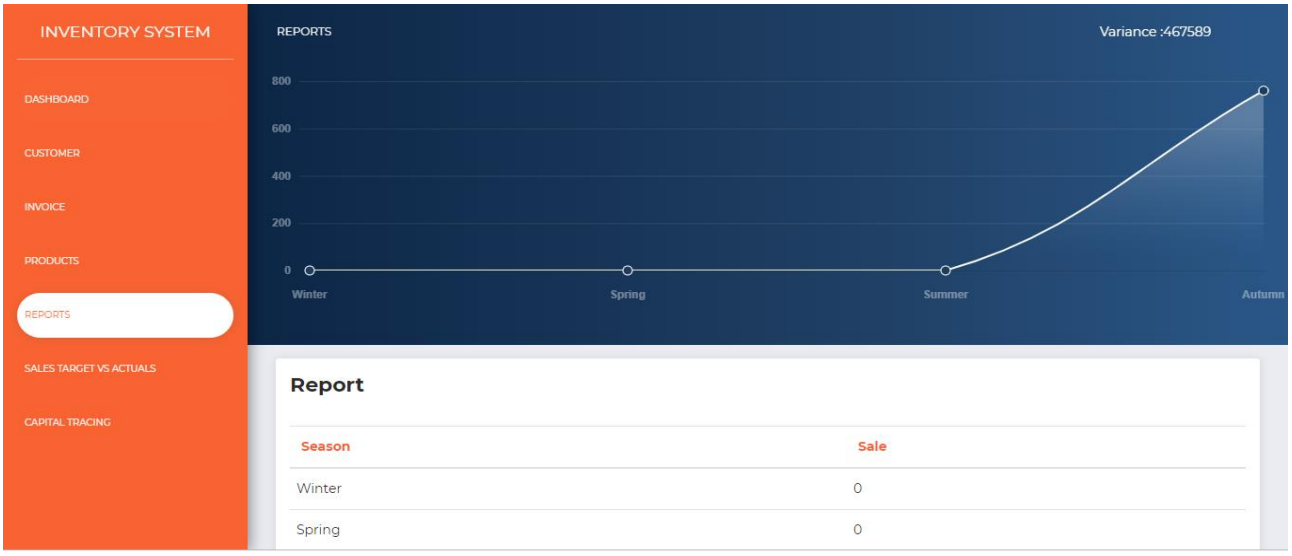
	var
▶	467589

5. Seasonal Trend:

The user enters the product ID **PID** and the year range – From **Y1** to **Y2** and clicks on submit button. The tabular and graphical representation of total sales for the given product season wise is displayed. The query gives monthly sales for the product and the front end groups the month season wise to find seasonal sales.

If user enters the range as 2018 to 2018 and PID as the query and corresponding output is as:

```
SELECT MONTH(SellDate) as month , SUM(Price) as total_sale
FROM sellsto AS s JOIN SALESINVOICE as si
WHERE YEAR(s.SellDate) BETWEEN 2018 AND 2018 AND si.Pid = 1
GROUP BY month
ORDER BY month;
```



	month	total_sale
▶	8	760
	9	2280
	10	9880
	11	2280

- ***Display the target sales Vs Actual Sales, Profit/Loss Monthly-using a graph. The target sales are entered by user for a given month & year:***

1. ***Inserting Sales Target:***

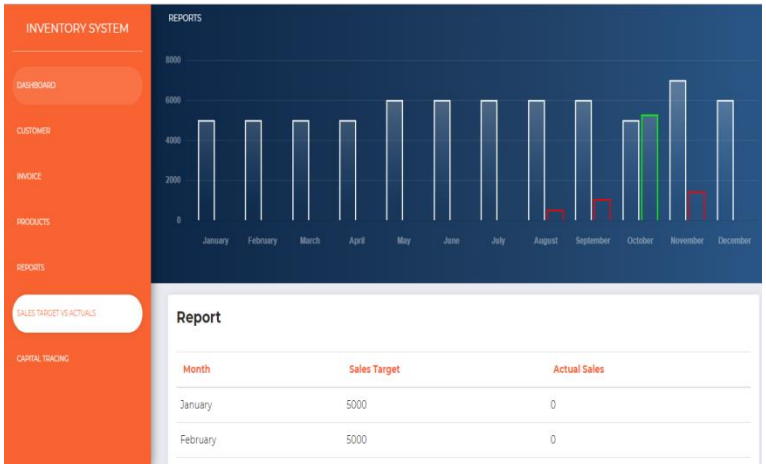
The user enters the month, year and his target Sales for that month and year. Suppose that user enters target sales as 7000 for November 2018. The corresponding Query is:

```
INSERT INTO salestarget VALUES(11,2018,7000,'user1')`;
```

2. ***Target Sales Vs Actual Sales:***

The user enters the year and his target Sales for each month of that year and the actual sales in that month is displayed graphically. Suppose that user enters year 2018. The corresponding Query and output is:

```
SELECT ActualSales.tmonth,target,actual from
   (select monthname(concat('2018-',Mnth,'-11')) AS tmonth,target
    from salestarget
    where Yr = 2018
    order by tmonth) AS TargetSales ,
   (SELECT monthname(SellDate) AS tmonth , SUM(Amount) AS actual
    FROM sellsto
    WHERE YEAR(SellDate) = 2018
    GROUP BY tmonth
    ORDER BY tmonth ) AS ActualSales
where TargetSales.tmonth=ActualSales.tmonth;
```



	tmonth	target	actual
▶	August	6000	525
	September	6000	1060
	October	5000	5275
	November	7000	1445

3. ***Monthly Profit/Loss:***

The user enters the year and his profit/loss % for each month of that year is displayed graphically. Suppose that user enters year 2018. The corresponding Query provides the front end with the sales_price and cost_price which then calculates the profit/loss. The output is:

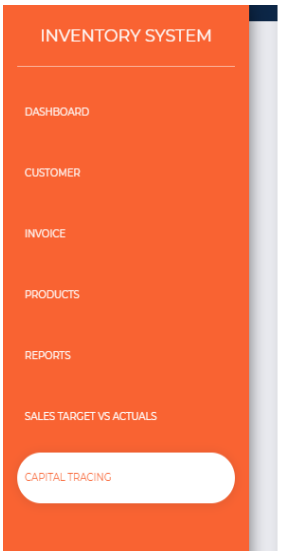
```
SELECT S.month,S.year, (sp-cp)*100/cp as profit from
(SELECT MONTH(SellDate) as month ,YEAR(SellDate) as year,SUM(Amount) AS sp
 FROM sellsto
 WHERE YEAR(SellDate) = 2018
 GROUP BY month,year
 ORDER BY month,year) AS S,

(SELECT month,year,SUM(cost_price) as cp
 FROM (SELECT MONTH(pb.PurDate) as month,YEAR(pb.PurDate) as
year,SUM(pi.Price) as cost_price
 FROM productinvoice as pi , purchasedby as pb
 WHERE pi.PInvNo = pb.PInvNo
 GROUP BY month,year
 UNION
 select Mnth as month,Yr as year,Production_cost as cost_price
 from produces) AS costTable
 WHERE year = 2018
 GROUP BY month,year
 ORDER BY month,year) AS C
 where S.month=C.month AND S.year=C.year;
```

	month	year	profit
▶	8	2018	-91.0103
	9	2018	-80.7273
	10	2018	1.4423
	11	2018	-71.6667

➢ ***Display the Capital Items Detail along with the current market Value:***

The Capital Items Details are displayed along with the current market value as per Rate of Depreciation in a tabular format. The Query and output is –

```
SELECT c.CPid,c.CPname,ci.Price,c.Quantity,c.RateDep,YEAR(pb.PurDate) as
Purchase_Year, (ci.price-((YEAR(sysdate())-
YEAR(pb.PurDate))*(c.RateDep)*(ci.price)/100)) as Current_Price
FROM capitalinvoice as ci, purchasedby as pb , capitalitems as c
WHERE ci.CPInvNo = pb.PInvNo  AND c.CPid = ci.CPid;
```



Simple Table

Capital ID	Capital Name	Price	Quantity	Rate of Depreciation	Purchase Year	Current Value
1	Freezer	20000	2	8	2018	20000
2	Centrifuge	16000	1	5	2018	16000
3	Cow	15000	4	null	2018	15000
4	Boiler	4000	1	7	2018	4000
5	Milk machine	4500	3	6.5	2018	4500
6	Computer	27000	1	20	2018	27000
7	Printer	1700	1	20	2018	1700
8	CCTV	3300	3	15	2018	3300

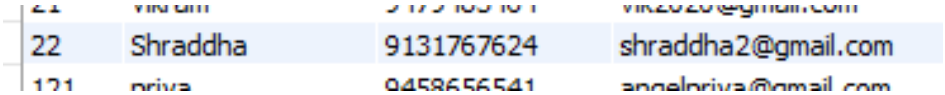
	CPid	CPname	Price	Quantity	RateDep	Purchase_Year	Current_Price
▶	1	Freezer	20000	2	8.00	2018	20000.000000
	2	Centrifuge	16000	1	5.00	2018	16000.000000
	3	Cow	15000	4	NULL	2018	NULL
	4	Boiler	4000	1	7.00	2018	4000.000000
	5	Milk machine	4500	3	6.50	2018	4500.000000
	6	Computer	27000	1	20.00	2018	27000.000000
	7	Printer	1700	1	20.00	2018	1700.000000
	8	CCTV	3300	3	15.00	2018	3300.000000

- ***Inserts And Updates.***

1. ***Inserting Customer:***

The user enters the Customer ID, Customer Name, Contact No. and E-mail ID. The corresponding Query for a sample data is:

```
INSERT INTO customer VALUES(22,'Shraddha',9131767624,'shraddha2@gmail.com');
```



22	Shraddha	9131767624	shraddha2@gmail.com

2. ***Inserting Suppliers:***

The user enters the Supplier ID, Supplier Name, Type, Contact No.. The corresponding Query for a sample data is:

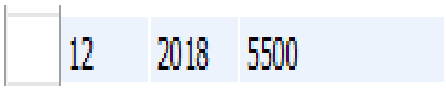
```
INSERT INTO supplier VALUES(14,'Gaurav','P',9595642151);
```

14	Gaurav	P	9595642151

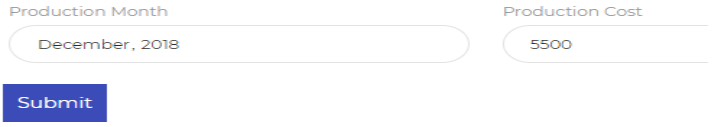
3. ***Inserting Production Cost:***

The user enters the Month, Year, Production Cost. The corresponding Query for a sample data is:

```
INSERT INTO produces VALUES(12,2018,5500);
```



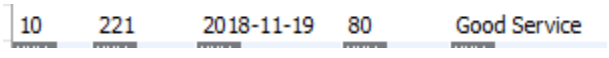
	12	2018	5500



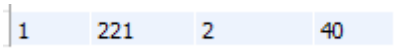
4. ***Inserting Invoice Details:***

The user enters the invoice details and front end calculates the bill. The corresponding Query for a sample data is:

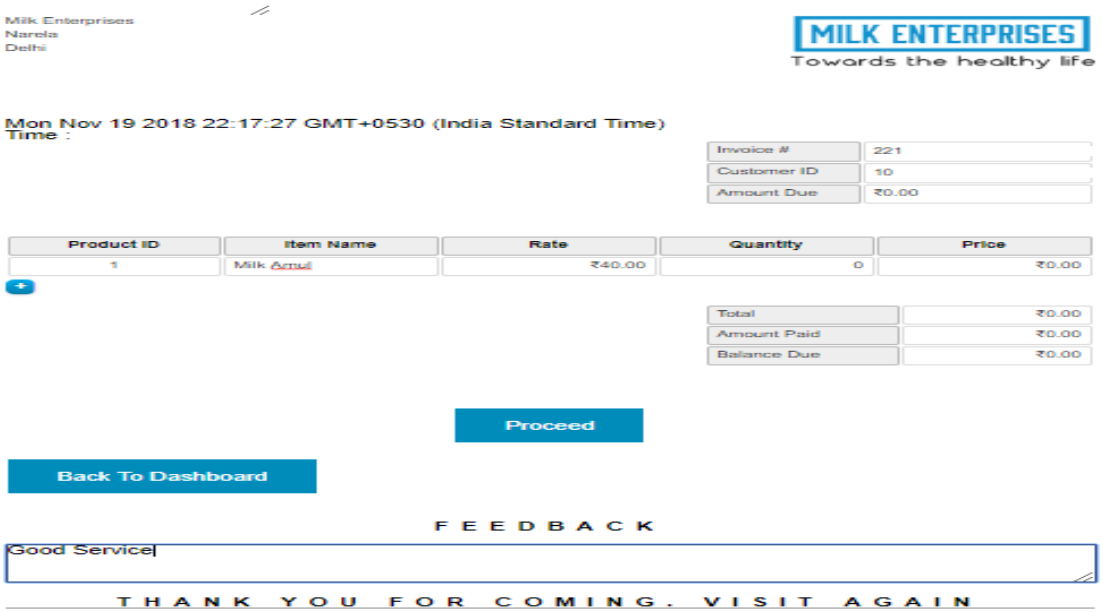
```
INSERT INTO sellsto VALUES(10,221,'2018-11-19',80,'Good Service');
INSERT INTO salesinvoice VALUES (1,221,2,40);
```



10	221	2018-11-19	80	Good Service



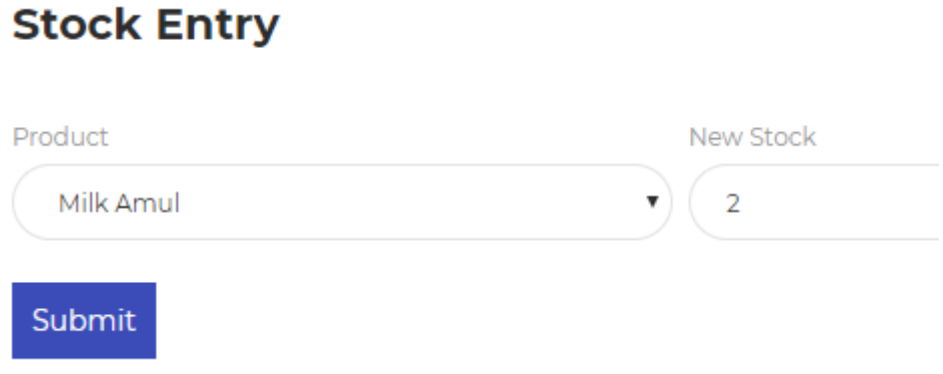
1	221	2	40



5. ***Updating Stock:***

The user selects the Product and enters the stock to be added. The corresponding Query for a sample data is:

```
UPDATE products
SET CurrentStock =  CurrentStock + 5
WHERE Pid = 1;
```



Pid	Pname	Price	CurrentStock
1	Milk Amul	40	56

Pid	Pname	Price	CurrentStock
1	Milk Amul	40	58

➢ *Display Customers and Products Details*

```
SELECT * FROM products;
```

Products

Products ID	Product Name	Price	Current Stock
1	Milk Amul	40	54
2	Ghee Amul	170	21
3	Ghee Ananda	140	15
4	Curd	20	50
5	Nasika	25	15
6	Ark1	35	15
7	Ark2	50	10
8	Ark3	55	10
9	Phenyl	100	20
10	Oil Mustard	120	30

Pid	Pname	Price	CurrentStock
1	Milk Amul	40	58
2	Ghee Amul	170	21
3	Ghee Ananda	140	15
4	Curd	20	50
5	Nasika	25	15
6	Ark1	35	15
7	Ark2	50	10
8	Ark3	55	10
9	Phenyl	100	20
10	Oil Mustard	120	30

```
SELECT * FROM customer;
```

Customer ID	Customer Name	Contact No	Email ID
1	John	7668123	john@gmail.com
2	Sam	1231569	sam@gmail.com
3	Todd	9766548	todd@gmail.com
4	Sara	6688457	sara@gmail.com
5	Frank	7889634	frank@outlook.com
6	Valerie	7894621	valerie@icloud.com
7	May	9847868	may@gmail.com
8	Martin	7694382	martin@outlook.com
9	Henry	5564865	henry@gmail.com
10	Kenny	6489987	kenny@gmail.com
11	Chester	6489951	chester@outlook.com
12	Mike	6841951	mike@icloud.com

Cid	Cname	ContactNo	EmailId
2	Sam	1231569	sam@gmail.com
3	Todd	9766548	todd@gmail.com
4	Sara	6688457	sara@gmail.com
5	Frank	7889634	frank@outlook.com
6	Valerie	7894621	valerie@icloud.com
7	May	9847868	may@gmail.com
8	Martin	7694382	martin@outlook.com
9	Henry	5564865	henry@gmail.com
10	Kenny	6489987	kenny@gmail.com
11	Chester	6489951	chester@outlook.com
12	Mike	6841951	mike@icloud.com

BIBLIOGRAPHY

- https://www.google.com/
- https://www.scholar.google.com/
- https://www.wikipedia.org/
- https://creately.com/ (used to create ER)

  It is a web-based diagramming tool with features like real-time collaboration.
- Oracle MySQL Community Server 8.0.13 (DBMS software)
- Nodejs – open-source, cross-platform JavaScript run-time environment that executes JavaScript code outside of a browser

  Libraries included – chartjs, popperjs, bootstrap
- Nodejs Express – Express is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications.
- Database System Concepts - Abraham Silberschatz, Henry F. Korth, S. Sudarshan, [6th edition]
- Fundamentals of Database Systems - Ramez Elmasri, Shamkant B. Navathe [6th edition]