Retail Management Database design
Final Project for Advanced Database Management System| Group 10| ISM6218.001F22.88140

Team Members

Akhil Ankireddypalli (U32430700)

Tejaswini (U34744827)

Sravani (U83404708)

Siva Sankari(U36650128)

Summary of Contents

Topic	Description	Team Member
Design	This phase covers logical database design which should be used for further implementation	Akhil, Tejaswini , Sravani , Siva Sankari
Implementation	Creating the objects which are mentioned in the design and dumping the data into tables	Akhil, Tejaswini , Sravani , Siva Sankari
Query Writing	Writing different queries to fetch the data effectively for various scenarios.	Akhil, Tejaswini , Sravani , Siva Sankari
Optimization and Other aspects	Performance Tuning, creating stored procs to achieve few tasks and creating different objects like views, sequences	Akhil, Tejaswini , Sravani , Siva Sankari

Objective

In this report, we describe the process that has been utilized to create a Database that could be used for a storage and management. We delve deep to describe the entities and attributes that have been tracked in the database. Moreover, this document explains the involved in query writing to fetch data, performance tuning using indexes, managing database integrity and data generation

Narrative

Nowadays, it is crucial to digitalize the information in order to have the competitive advantage of the data for all inventory management enterprises in building better customer relations, better understanding of the logistics by having all the Object information on hand, and for easier management.

The Retail management database has a total of 8 tables which includes the Object Class which has the class id, class name, and object description like object id and foreign keys like customer id and distributor id. The class table has the information used to categorize the objects.

The customer information table gives information such as CustomerId, and other attributes such as Name, Address, phone number. Distributor information table has the attributes like DistributorId and Distributor Name which gives us the details of the supplier

Entities Identified to be Tracked

• Catalogue

• Class

• Consumed_object

• Customer_info

• Object

• Distributor_information

• Payment

• Purchase_detail

Entities with Attributes Nested

• Catalogue

- ObjectID (Primary key)
- USD

• Class

- ClassID (primary key)
- ClassName

• Distributer_information

- DistributerID(PK)
- DistributerName
- Address
- Phone

• payment

- PurchaseID(PK)
- Money_Paid
- Payment_Mode
- Payment_Date

• object

- ObjectID(PK)
- Object_name

- ClassID(FK1)
- DistributerID(FK2)
- In_stock
- Unit_Price
- ReorderLevel

• consumed_object

- ObjectID(PK)
- Amount

• customer_info

- CustomerID(PK)
- Name
- Address
- PhoneNo
- Password

• purchase_detail

- PurchaseID(PK)
- ObjectID(FK)
- CustomerID(FK)
- Quantity
- Discount
- Total_Amount
- Trans_Init_Date

Business Rules

- One object have connection to a single PurchaseID and ObjectID.
- Customer can order multiple purchaseIDs, but one product can only be mapped to a single customer.
- One Distributer can have multiple objects under him, and the objects can also have multiple Distributers.

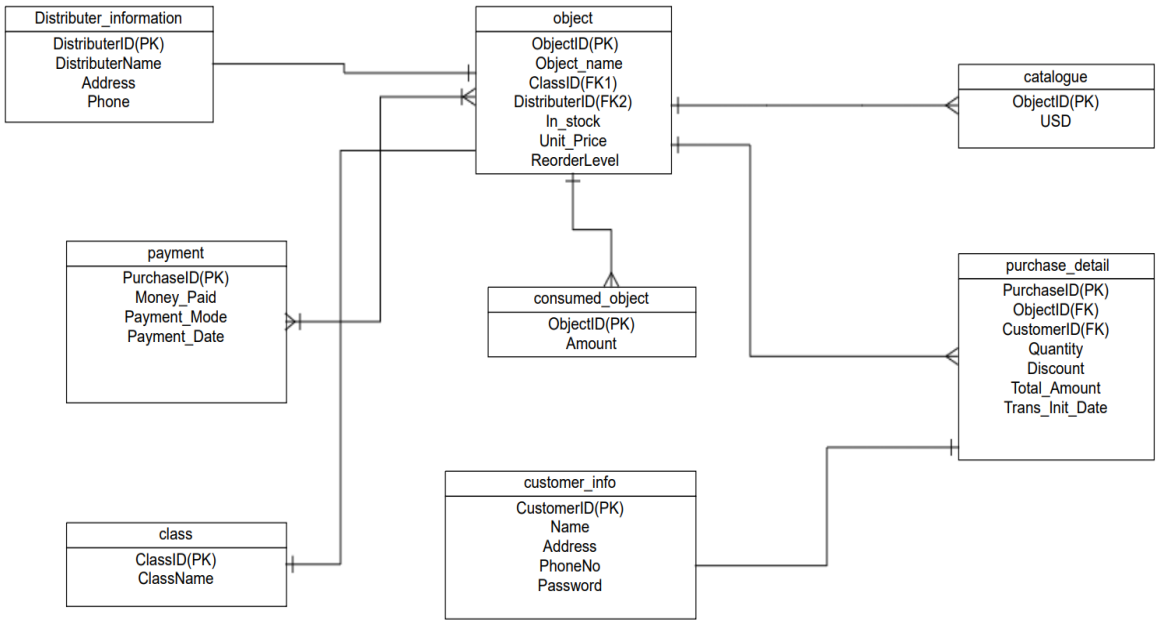
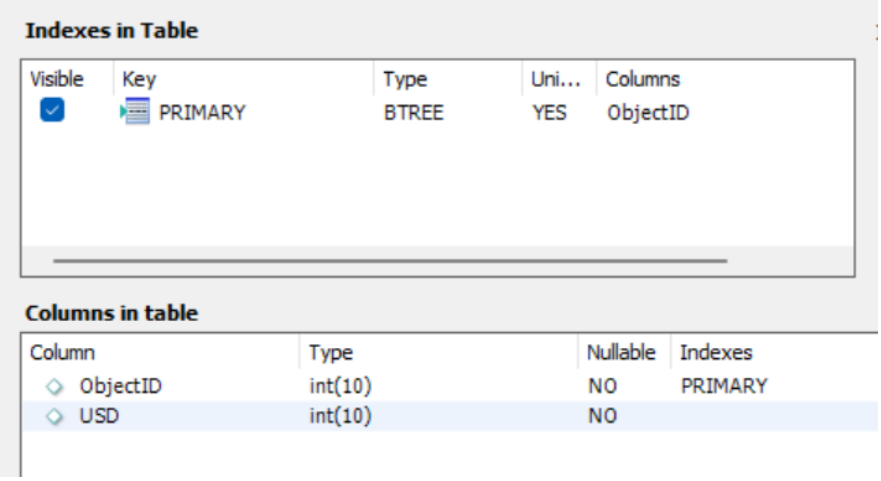


Table Views

Catalogue Table

This table collects the all details about ObjectID and USD



Class Table

the table consists of ClassID and ClassName

Indexes in Table

Visible	Key	Type	Uni...	Columns
☑	PRIMARY	BTREE	YES	ClassID

Columns in table

Column	Type	Nullable	Indexes
ClassID	int(20)	NO	PRIMARY
ClassName	varchar(30)	NO	

consumed_object Table

the table consists of ObjectID and Amount.

Indexes in Table

Visible	Key	Type	Uni...	Columns
☑	PRIMARY	BTREE	YES	ObjectID

Columns in table

Column	Type	Nullable	Indexes
ObjectID	int(10)	NO	PRIMARY
Amount	int(10)	NO	

customer_info Table

The table consists of CustomerID, Name, Address, PhoneNo and Password.

Indexes in Table

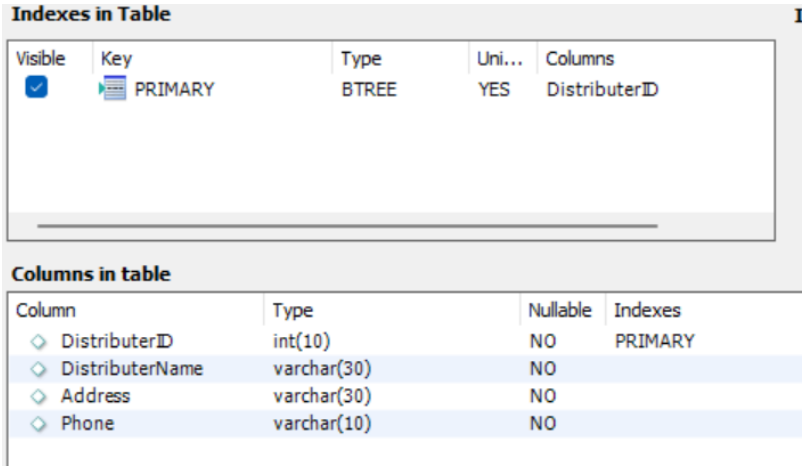
Visible	Key	Type	Uni...	Columns
☑	PRIMARY	BTREE	YES	CustomerID

Columns in table

Column	Type	Nullable	Indexes
CustomerID	int(11)	NO	PRIMARY
Name	varchar(30)	NO	
Address	varchar(50)	NO	
PhoneNo	varchar(15)	NO	
Password	varchar(15)	NO	

Distributer_information Table

The table consists of DistributerID, DistributerName, Address and Phone.



Indexes in Table

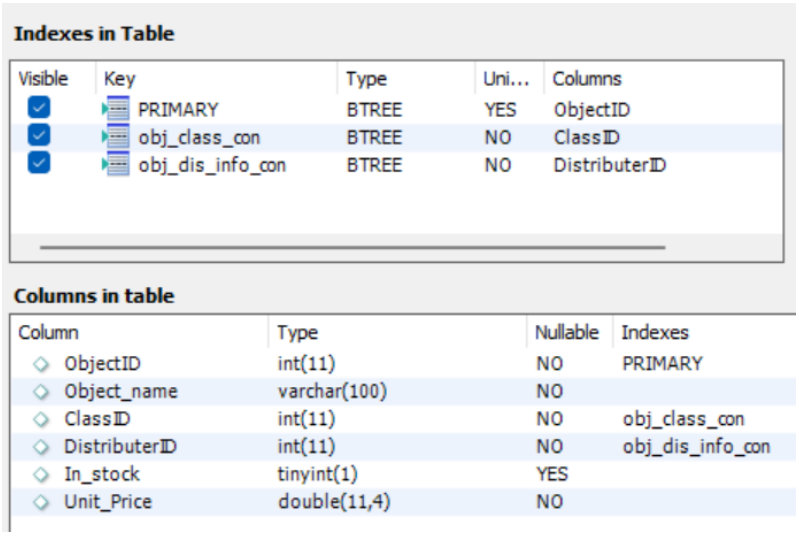
Visible	Key	Type	Uni...	Columns
☑	PRIMARY	BTREE	YES	DistributerID

Columns in table

Column	Type	Nullable	Indexes
DistributerID	int(10)	NO	PRIMARY
DistributerName	varchar(30)	NO	
Address	varchar(30)	NO	
Phone	varchar(10)	NO	

Object Table

The table consists of ObjectID, Object_Name, ClassID, DistributerID, In-stock and Unit_price



Indexes in Table

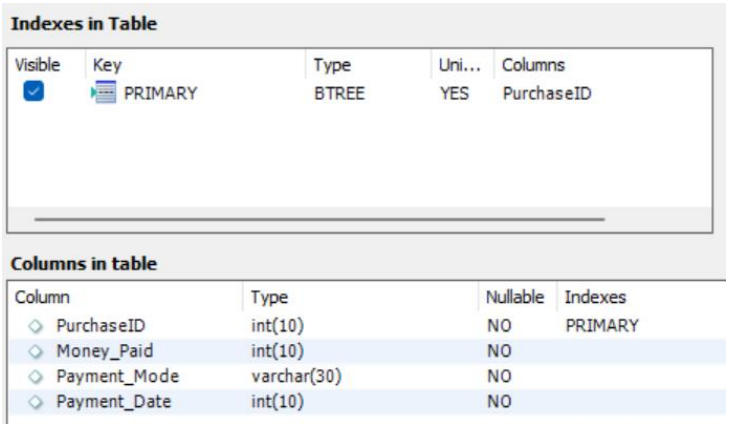
Visible	Key	Type	Uni...	Columns
☑	PRIMARY	BTREE	YES	ObjectID
☑	obj_class_con	BTREE	NO	ClassID
☑	obj_dis_info_con	BTREE	NO	DistributerID

Columns in table

Column	Type	Nullable	Indexes
ObjectID	int(11)	NO	PRIMARY
Object_name	varchar(100)	NO	
ClassID	int(11)	NO	obj_class_con
DistributerID	int(11)	NO	obj_dis_info_con
In_stock	tinyint(1)	YES	
Unit_Price	double(11,4)	NO	

Payment Table

The table consists of PurchaseID, Money_Paid, Payement_Mode and Payment_date



Indexes in Table

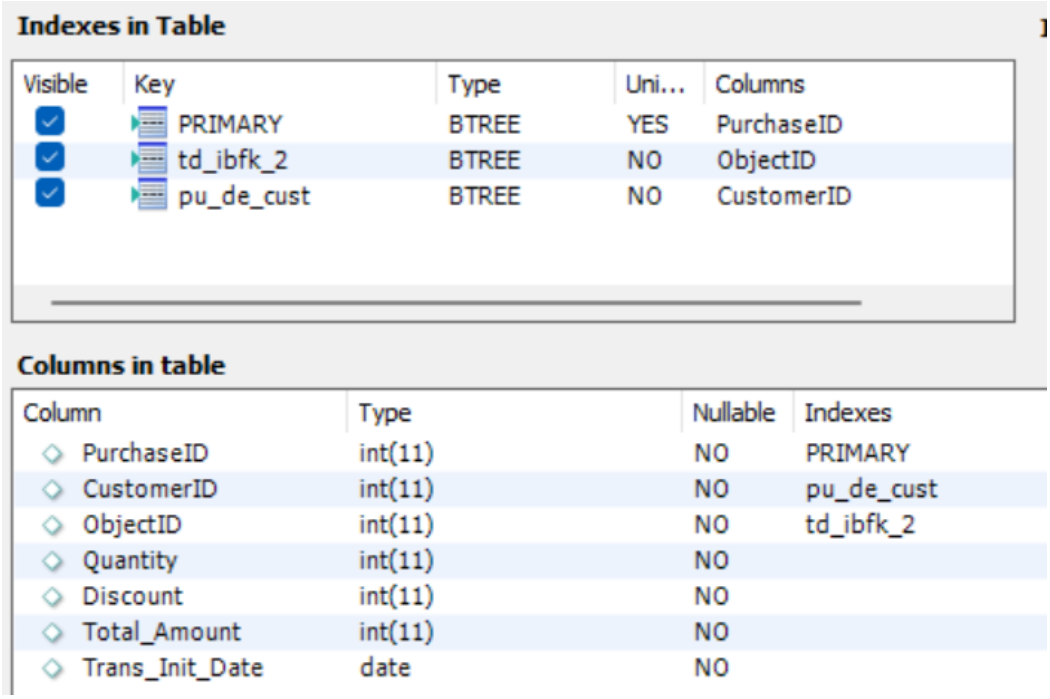
Visible	Key	Type	Uni...	Columns
☑	PRIMARY	BTREE	YES	PurchaseID

Columns in table

Column	Type	Nullable	Indexes
PurchaseID	int(10)	NO	PRIMARY
Money_Paid	int(10)	NO	
Payment_Mode	varchar(30)	NO	
Payment_Date	int(10)	NO	

Purchase_detail Table

The table consists of PurchaseID, CustomerID, ObjectID, Quantity, Discount, Total_amount and Trans_init_Date



Indexes in Table

Visible	Key	Type	Uni...	Columns
☑	PRIMARY	BTREE	YES	PurchaseID
☑	td_ibfk_2	BTREE	NO	ObjectID
☑	pu_de_cust	BTREE	NO	CustomerID

Columns in table

Column	Type	Nullable	Indexes
PurchaseID	int(11)	NO	PRIMARY
CustomerID	int(11)	NO	pu_de_cust
ObjectID	int(11)	NO	td_ibfk_2
Quantity	int(11)	NO	
Discount	int(11)	NO	
Total_Amount	int(11)	NO	
Trans_Init_Date	date	NO	

Script used for designing the database.

Creating table for Catalouge

```
CREATE TABLE `catalogue`
(
 `ObjectID` int(10) NOT NULL,
 `USD` int(10) NOT NULL,
 PRIMARY KEY (`ObjectID`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Creating table for Class

```
CREATE TABLE `class`
 (
 `ClassID` int(20) NOT NULL,
 `ClassName` varchar(30) NOT NULL,
 PRIMARY KEY (`ClassID`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Creating table for consumed_object

```
CREATE TABLE `consumed_object`
(
 `ObjectID` int(10) NOT NULL,
```

```
  `Amount` int(10) NOT NULL,
  PRIMARY KEY (`ObjectID`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Creating table for customer_info

```
CREATE TABLE `customer_info`
 (
  `CustomerID` int(11) NOT NULL,
  `Name` varchar(30) NOT NULL,
  `Address` varchar(50) NOT NULL,
  `PhoneNo` varchar(15) NOT NULL,
  `Password` varchar(15) NOT NULL,
  PRIMARY KEY (`CustomerID`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Creating Table for object

```
CREATE TABLE `object`
 (
  `ObjectID` int(11) NOT NULL,
  `Object_name` varchar(100) NOT NULL,
  `ClassID` int(11) NOT NULL,
  `DistributerID` int(11) NOT NULL,
  `In_stock` tinyint(1) DEFAULT '1',
  `Unit_Price` double(11,4) NOT NULL,
  PRIMARY KEY (`ObjectID`),
  KEY `obj_class_con` (`ClassID`),
  KEY `obj_dis_info_con` (`DistributerID`),
  CONSTRAINT `obj_class_con` FOREIGN KEY (`ClassID`) REFERENCES `class` (`ClassID`),
  CONSTRAINT `obj_dis_info_con` FOREIGN KEY (`DistributerID`) REFERENCES
`distributer_information` (`DistributerID`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

Creating table for distributer_information

```
CREATE TABLE `distributer_information`
 (
  `DistributerID` int(10) NOT NULL,
  `DistributerName` varchar(30) NOT NULL,
  `Address` varchar(30) NOT NULL,
  `Phone` varchar(10) NOT NULL,
  PRIMARY KEY (`DistributerID`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Creating table for object

```
CREATE TABLE `object`
 (
  `ObjectID` int(11) NOT NULL,
```

```
  `Object_name` varchar(100) NOT NULL,
  `ClassID` int(11) NOT NULL,
  `DistributerID` int(11) NOT NULL,
  `In_stock` tinyint(1) DEFAULT '1',
  `Unit_Price` double(11,4) NOT NULL,
  PRIMARY KEY (`ObjectID`),
  KEY `obj_class_con` (`ClassID`),
  KEY `obj_dis_info_con` (`DistributerID`),
  CONSTRAINT `obj_class_con` FOREIGN KEY (`ClassID`) REFERENCES `class` (`ClassID`),
  CONSTRAINT `obj_dis_info_con` FOREIGN KEY (`DistributerID`) REFERENCES
`distributer_information` (`DistributerID`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

Creating table for payment

```
CREATE TABLE `payment`
 (
  `PurchaseID` int(10) NOT NULL,
  `Money_Paid` int(10) NOT NULL,
  `Payment_Mode` varchar(30) NOT NULL,
  `Payment_Date` int(10) NOT NULL,
  PRIMARY KEY (`PurchaseID`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Creating table for purchase_detail

```
CREATE TABLE `purchase_detail`
 (
  `PurchaseID` int(11) NOT NULL,
  `CustomerID` int(11) NOT NULL,
  `ObjectID` int(11) NOT NULL,
  `Quantity` int(11) NOT NULL,
  `Discount` int(11) NOT NULL DEFAULT '0',
  `Total_Amount` int(11) NOT NULL,
  `Trans_Init_Date` date NOT NULL,
  PRIMARY KEY (`PurchaseID`),
  KEY `td_ibfk_2` (`ObjectID`),
  KEY `pu_de_cust` (`CustomerID`),
  CONSTRAINT `pu_de_cust` FOREIGN KEY (`CustomerID`) REFERENCES `customer_info`
(`CustomerID`),
  CONSTRAINT `td_ibfk_2` FOREIGN KEY (`ObjectID`) REFERENCES `object` (`ObjectID`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Queries

Finding the customer who spent the highest amount, total amount spent by the customer and the object bought

```
select c.Name, o.Object_name,max(p.Total_Amount)
from category.customer_info c
join category.purchase_detail p
    on c.CustomerID=p.CustomerID
join category.object o
    on o.ObjectID=p.ObjectID
```

	Name	Object_name	max(p.Total_Amount)
▶	Elizabeth Windsor	Wine - Rosso Toscano Igt	3000

We updated payment table for different purchase IDs

```
UPDATE category.payment set PurchaseID =23 WHERE Money_Paid =940;

update category.payment set PurchaseID =25 where Money_Paid =923;
update category.payment set PurchaseID =26 where Money_Paid =879;
update category.payment set PurchaseID =27 where Money_Paid =780;
update category.payment set PurchaseID =28 where Money_Paid =459;
update category.payment set PurchaseID =29 where Money_Paid =287;
update category.payment set PurchaseID =30 where Money_Paid =245;
```

✓	47	21:16:51	UPDATE category.payment set PurchaseID =23 WHERE Money_Paid =940	0 row(s) affected Rows matched: 1 Changed: 0 Warnings: 0	0.016 sec
✓	48	21:16:52	update category.payment set PurchaseID =25 where Money_Paid =923	0 row(s) affected Rows matched: 1 Changed: 0 Warnings: 0	0.000 sec
✓	49	21:16:52	update category.payment set PurchaseID =26 where Money_Paid =879	0 row(s) affected Rows matched: 1 Changed: 0 Warnings: 0	0.000 sec
✓	50	21:16:52	update category.payment set PurchaseID =27 where Money_Paid =780	0 row(s) affected Rows matched: 1 Changed: 0 Warnings: 0	0.000 sec
✓	51	21:16:52	update category.payment set PurchaseID =28 where Money_Paid =459	0 row(s) affected Rows matched: 1 Changed: 0 Warnings: 0	0.000 sec
✓	52	21:16:52	update category.payment set PurchaseID =29 where Money_Paid =287	0 row(s) affected Rows matched: 1 Changed: 0 Warnings: 0	0.000 sec
✓	53	21:16:52	update category.payment set PurchaseID =30 where Money_Paid =245	0 row(s) affected Rows matched: 1 Changed: 0 Warnings: 0	0.000 sec

We created a procedure to get the total amount for different distributers based on ObjectID.

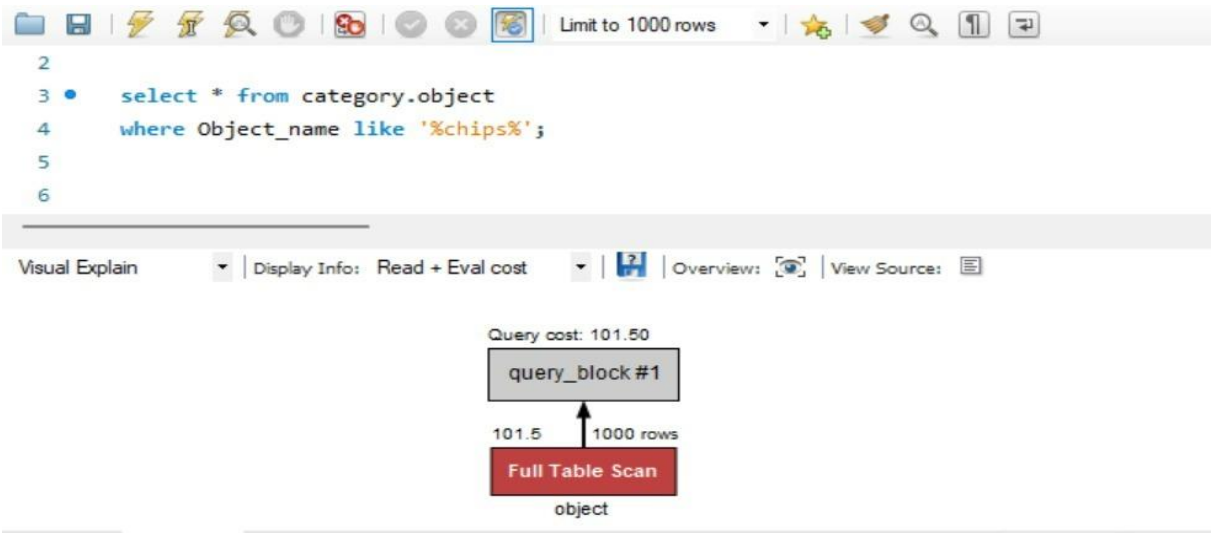
```
DELIMITER //
create procedure category.summary (objectid int)
begin
select d.DistributerName,o.ObjectID,sum(p.Total_Amount)
from category.purchase_detail p
join category.object o on p.ObjectID = o.ObjectID
join category.distributer_information d on d.DistributerID = o.DistributerID
where o.ObjectID = objectid
group by 1,2;

end
//

DELIMITER ;

call category.summary (1);
```

Performance tuning

To improve the performance of our database, we have used Indexing

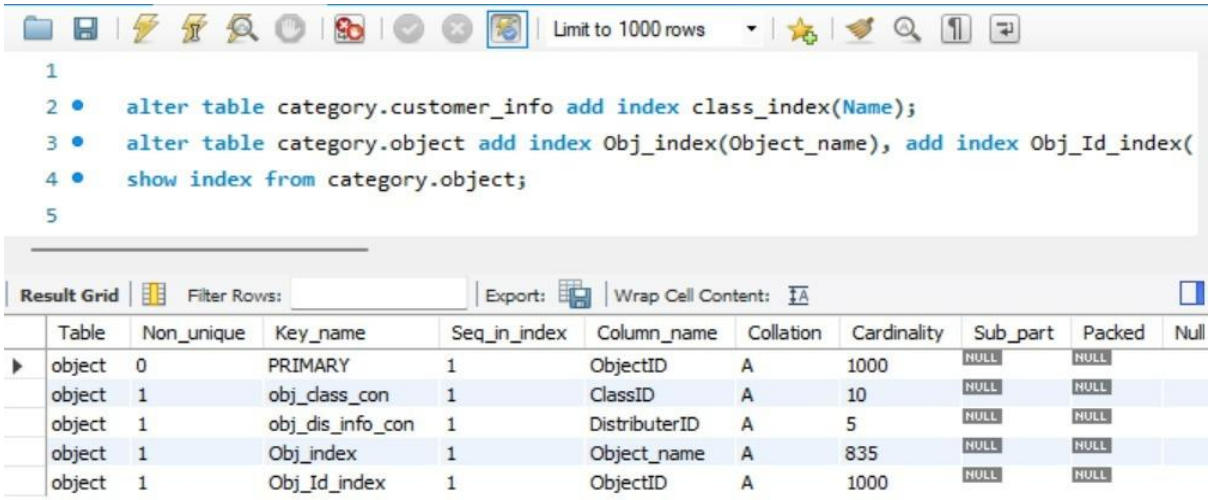
Indexing is used to increase the speed of retrieval of required information thereby decreasing the runtime and cost of query. This is accomplished by reducing the number of data instances to be reviewed with indexes, the every instance need not be perused to retrieve the required data

- Before adding index to the tables



Here, we can see the cost of query to be 101.5

- Adding index for the table 'object' with the names Obj_index and Obj_Id_index on the columns Object_name and Object_id
- We have added index for the table Customer_info on the column ‘Name’

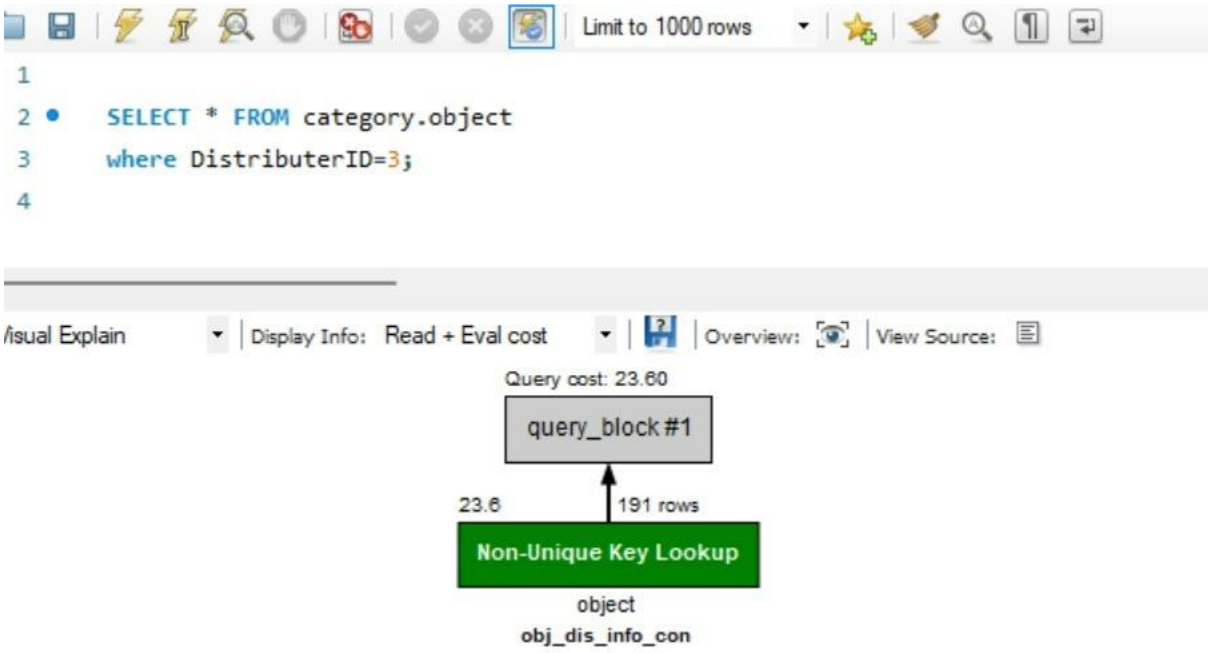


```
1
2 •  alter table category.customer_info add index class_index(Name);
3 •  alter table category.object add index Obj_index(Object_name), add index Obj_Id_index(
4 •  show index from category.object;
5
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null
object	0	PRIMARY	1	ObjectID	A	1000	NULL	NULL	
object	1	obj_class_con	1	ClassID	A	10	NULL	NULL	
object	1	obj_dis_info_con	1	DistributerID	A	5	NULL	NULL	
object	1	Obj_index	1	Object_name	A	835	NULL	NULL	
object	1	Obj_Id_index	1	ObjectID	A	1000	NULL	NULL	

- Running the queries after indexing



```
1
2 •  SELECT * FROM category.object
3    where DistributerID=3;
4
```

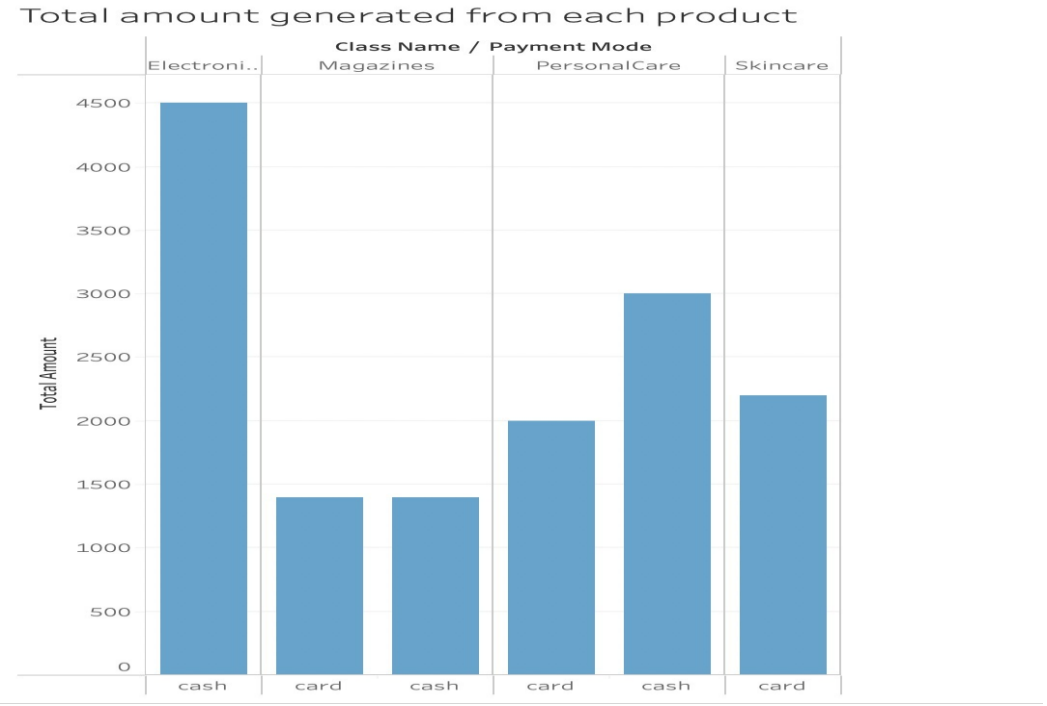
Visual Explain | Display Info: Read + Eval cost | Overview: | View Source:

- After indexing, we can see that there is a considerable decrease in the query cost in both queries which are 23.6 and 20.9 respectively
- From this, we can prove that introducing indexed into the tables increases the performance of database
- Using indexing, we increased the performance of our database

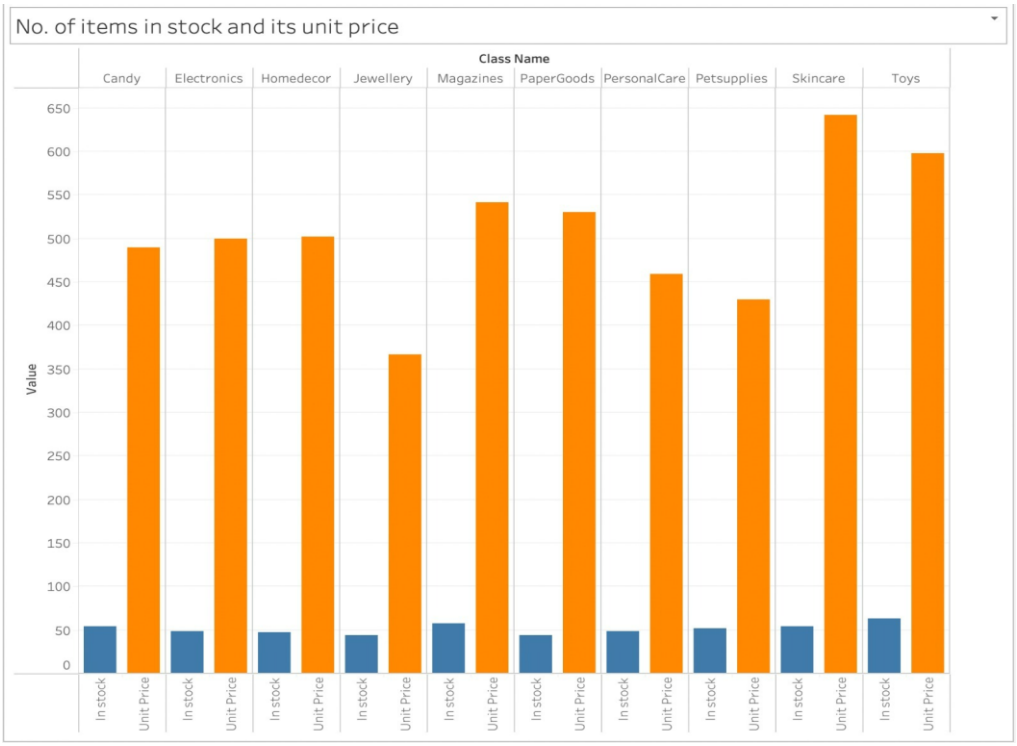
DATA VISUALISATION

We have performed data visualization to get output for some queries

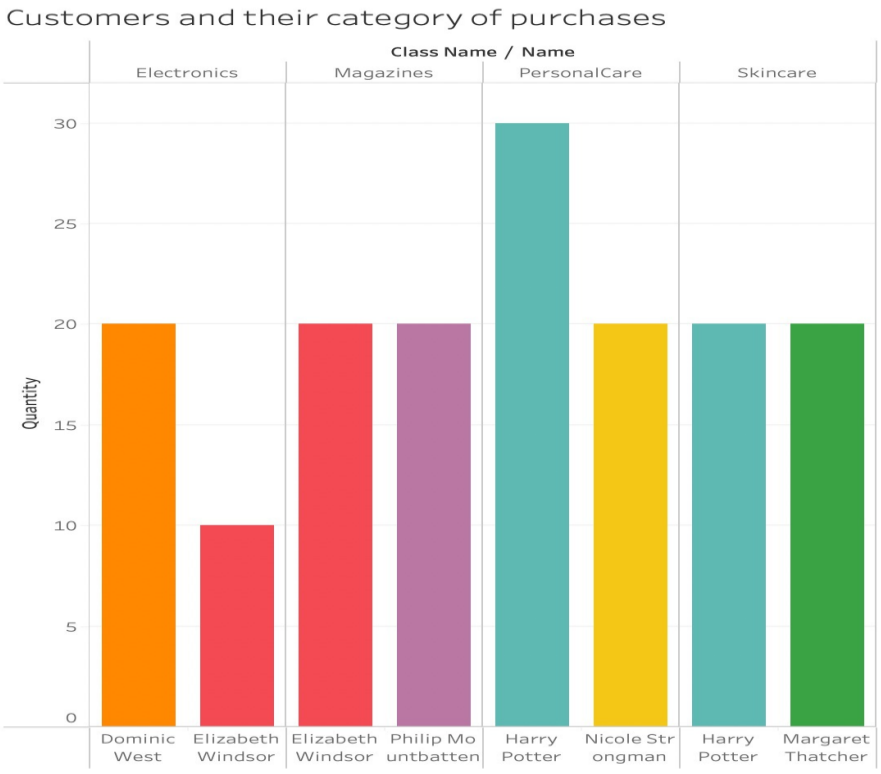
- which category generated highest revenue



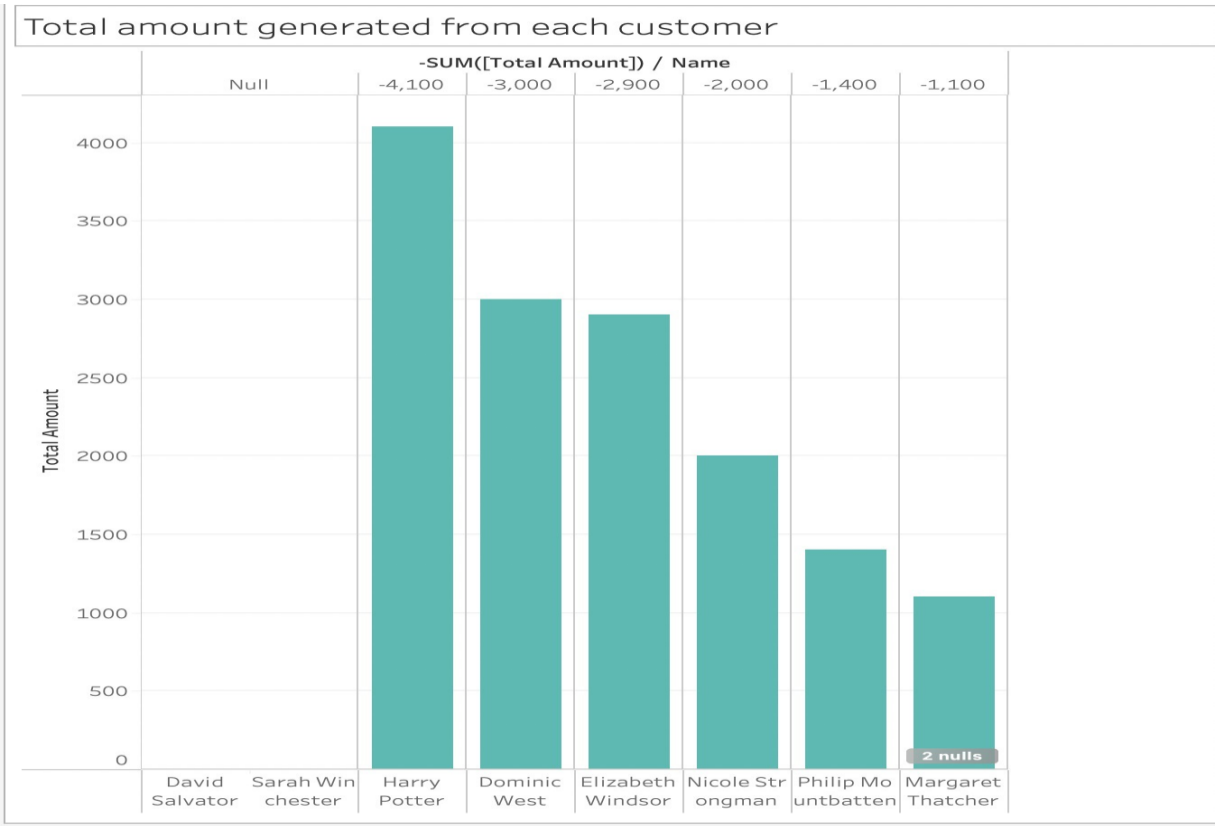
- How many items are in stock and its unit price?



- What category of products purchased by each customer?



- who generated maximum revenue?



- Who are the distributors for each category and its unit price?