# IIITD Furniture Store

# **Team Xypnix**: Shiva (2017190), Sunny(2016270), Ashwani(2016232), Ayush(2017286), Vishal(2016278)

**Goal:** The goal of this project is to develop the back-end of an online store with extensive use of data entities selection and the relationship between them, data modeling, database access, and data manipulation.

**Description:** Our team plan to start an Online Furniture store that will provide a platform to seller and renters to register their Furniture products and buyers to browse the products. Admin adds or deletes the selling and renting stores after their request. Admin are us i.e. founders of the store.

The following are the stakeholders:

* Buyers
* Renters
* Sellers
* Shipping Company
* Admin

### **Workload:**

### **Costumer:**

* Search Furniture products by categories and price?
* Who are retailers available in the city?
* What are the Contact Details of the store ?.
* Check Order List.
* Which products are on offer.

### **Shipping Company:**

* What are the retailer’s contact and address?
* What are the buyer’s contact and address?
* How many products are needed to be delivered?
* What is the total cost of the delivery?
* Details of the delivery person.
* Assign delivery person.

### **Seller:**

* How many products left in stock?
* How many people have ordered the products?
* What are the customer’s address and contact no. ?
* What is the total price?

### **Renter:**

* How many products left in stock?
* How much rent is due?
* What duration is left for the rented product?
* What are the customer’s address and contact no. ?
* How many people have ordered the products?

### **Admin:**

* Accepting(or rejecting) the request of a person as seller, renters, and the shipping company and creating their database.
* Creating a buyer database(id, address, contact information).
* Adding(or deleting) another admin.
* Adding privileges of buyer, seller, renter, shipping company.
* Will have a track of sales.

# Database Schema

**Seller***( S\_ID: int , S\_name: String, S\_contact: int, S\_city: String, S\_address: String, S\_Zip Code: int, primary key( S\_ID), );*

**Renter***( R\_ID: int, R\_name: String, R\_contact: int, R\_city: String, R\_address: String, R\_Zip Code: int , primary key(S\_ID))*;

**Sale Product** *( SP\_ID: int, RP\_name: String, SP\_Category\_ID: Int, S\_ID: int, SP\_Price:int , primary key (SP\_ID) , foreign key(S\_ID) refrences Seller, foreign key(Category\_ID) refrences Category);*

**Rent Product** *( RP\_ID: int, RP\_name: String, RP\_Category\_ ID: Int, R\_ID:int, RP\_Price:int, primary key (RP\_ID), foreign key(R\_ID) refrences Renter, foreign key(Category\_ID) refrences Category) );*

**Shipping Company** *( Ship\_ID: int; Del\_Person\_ID: int, Name: String, Contact: int, primary key(Ship\_ID), foreign key(Del\_Person\_ID) refrences Delivery Person));*

**Customer** *( Cust\_ID: Int, Cust\_name: String, Cust\_address: String, Cust\_state: String, Cust\_City: String, Cust\_Zip:int, Primary Key(Cust\_ID , primary key(Cust\_ID));*

**Delivery Person** *( Order\_ID; int, Del\_Person\_ID: int; DelPerson\_name: String, Contact: int, primary key(Del\_Person\_ID), foreign key(Order\_ID) refrences Sales Oder, Rent Orders) ;*

**Category** *( Category\_ID: int, Category\_name: String, primary key(Category\_ID));*

**Sale Orders** *( Order\_ID: int, SP\_ID: int, Customer\_ID: int, Shipper\_ID: int, Price: int, Order\_date: date, Shipping Date: date,primary key(Order\_ID), foreign key(SP\_ID) refrences Sale Product, foreign key(Shipper\_ID) refrences Shipping Company) );*

**Rent Orders** *( Order\_ID: int, RP\_ID: int, Customer\_ID: int, Shipper\_ID: int, Rent: int, Order\_date: date, Shipping Date: date, ReturnDate: date, primary key( Order\_ID) , foreign key(RP\_ID) refrences Rent Product, foreign key(Shipper\_ID) refrences Shipping Company));*

**Admin***( Admin\_ID: int, Request\_ID: int, Admin\_name: string, Admin\_mail: String, primary key(Admin\_ID) , foreign key(Request\_ID) refrences Request);*

**Request** *( Request\_ID: int, Person\_name: String, Person\_contact: String, Contact\_Type: String,primary key(Request\_ID));*

***Offerpricesaleprod****(SP\_ID int,Off int, Primary Key (SP\_ID));*

***Offerpricerentprod****(RP\_ID int, Off int, Primary Key (SP\_ID));*

**Reach**( *username varchar(100),email varchar(100),contact bigint,comments longtext* );

## **Queries**

### **Indexes**

We have already made crucial attributes as primary key and foreign key to save up the cost. From the rest of the attributes, we have made the following indexes.

**Indexes for SP\_name, RP\_name ( selling products and rent products name)**

Since the cost for searching the selling products name and rent products name was high we created indexes on them to save the cost and memory. Using where clause was searching total 73 rows, but when an index was created on the query, searching become efficient only took the number of rows where our interest of result was. Like searching ‘arm chair’ only took 10 rows to search instead of 73 rows for our database.

*EXPLAIN SELECT SP\_ID, S\_ID, SP\_name from saleproducts where SP\_name= 'arm chair';*

*CREATE INDEX SP\_name ON saleproducts(SP\_name);*

*show index from saleproducts;*

*EXPLAIN SELECT SP\_ID, S\_ID from saleproducts where SP\_name= 'arm chair';*

*EXPLAIN SELECT RP\_ID, R\_ID, RP\_name from rentproduct where RP\_name= 'arm chair';*

*CREATE INDEX RP\_name ON rentproduct(RP\_name);*

*show index from rentproduct;*

*EXPLAIN SELECT RP\_ID, R\_ID, RP\_name from rentproduct where RP\_name= 'arm chair';*

**Indexes for seller or renter’s city.**

While searching for the seller or renter’s city for the products, result was generated by searching through all rows, but when the index was created on the city, result only came up by searching only the city in which the customer is interested. For example, searching a seller in New Delhi took 13 rows with Where clause. While searching with the index it took only 3 rows for our database.

*Explain SELECT R\_ID, R\_name from renter where R\_city= 'New Delhi';*

*CREATE INDEX R\_city ON renter(R\_city);*

*show index from renter;*

*Explain SELECT R\_ID, R\_name from renter where R\_city= 'New Delhi';*

*Explain SELECT S\_ID, S\_name from seller where S\_city= 'New Delhi';*

*CREATE INDEX S\_city ON seller(S\_city);*

*show index from seller;*

*Explain SELECT S\_ID, S\_name from seller where S\_city= 'New Delhi';*

**Indexes for prices**

Since there are a lot of products in our database, searching their price especially between some range was slow. So, we created indexes for price on both sale products and rent products. As a result, the products whose selling prices were between 500 and 700 took only 11 rows for result which were taking 73 rows earlier.

*Explain select Sp\_ID, S\_ID from saleproducts where SP\_price between 500 and 700;*

*CREATE INDEX Price ON saleproducts( SP\_Price);*

*show indexes from saleproducts;*

*Explain select Sp\_ID, S\_ID from saleproducts where SP\_price between 500 and 700;*

*Explain select Rp\_ID, R\_ID from saleproducts where RP\_price between 500 and 700;*

*CREATE INDEX Price ON rentproduct( RP\_Price);*

*show indexes from rentproduct;*

*Explain select Rp\_ID, R\_ID from rentproduct where RP\_price between 500 and 700;*

**Alter**

Initially, we mentioned order dates in date time format but later we realized that we only need date. So, we made alteration to our type of order date attribute using alter command.

ALTER TABLE salesorders MODIFY order\_date date;

ALTER TABLE salesorders MODIFY shipping\_date date;

ALTER TABLE rentorders MODIFY order\_date date;

ALTER TABLE rentorders MODIFY shipping\_date date;

ALTER TABLE rentorders MODIFY return\_date date;

**Relational Queries**

**Find the product name, category and price by a seller name**

select Sp\_name,Category\_Id, Sp\_price from saleproducts natural join seller where seller.s\_name="Galaxy Wooden Designer";

**Find the product name, category and price by a renter name**

select Rp\_name,Category\_Id, Rp\_price from rentproduct natural join renter where renter.r\_name="Prince Furniture";

**Find the seller who sells either two of three types of products and list the names of the products**

*select S1.S\_Id, S1.S\_name as Seller, P.SP\_name as Product, S1.S\_contact as Contact, S1.S\_city as city ,S1.S\_address as Address ,S1.S\_Zipcode as Zipcode from seller S1 natural join saleproducts P where P.category\_Id= 3 or P.category\_ID =4;*

*select S1.S\_Id, S1.S\_name as Seller, P.SP\_name as Product, S1.S\_contact as Contact, S1.S\_city as city ,S1.S\_address as Address ,S1.S\_Zipcode as Zipcode from seller S1 natural join saleproducts P where P.category\_Id= 1 or P.category\_ID =4;*

*select S1.S\_Id, S1.S\_name as Seller, P.SP\_name as Product, S1.S\_contact as Contact, S1.S\_city as city ,S1.S\_address as Address ,S1.S\_Zipcode as Zipcode from seller S1 natural join saleproducts P where P.category\_Id= 1 or P.category\_ID =3;*

**Find the renter who either two of three types of products on rent and list the names of the products**

*Select R1.R\_Id, R1.R\_name as Renter, P.RP\_name as Product, R1.R\_contact as Contact, R1.R\_city as city ,R1.R\_address as Address ,R1.R\_Zipcode as Zipcode from renter R1 natural join rentproduct P where P.category\_Id= 3 or P.category\_ID =4;*

*Select R1.R\_Id, R1.R\_name as Renter, P.RP\_name as Product, R1.R\_contact as Contact, R1.R\_city as city ,R1.R\_address as Address ,R1.R\_Zipcode as Zipcode from renter R1 natural join rentproduct P where P.category\_Id= 1 or P.category\_ID =4;*

*Select R1.R\_Id, R1.R\_name as Renter, P.RP\_name as Product, R1.R\_contact as Contact, R1.R\_city as city ,R1.R\_address as Address ,R1.R\_Zipcode as Zipcode from renter R1 natural join rentproduct P where P.category\_Id= 1 or P.category\_ID =3;*

**Find the avg sale of a given month**

Example- Month 2

*select avg(quantity\*saleprice(S.SP\_ID)) as av from salesorders S where month(S.order\_date) between 2 and 3;*

**Find the rent of a given month**

Example- Month 2

*select avg(quantity\*rentprice(S.RP\_ID)) as av from rentorders S where month(S.order\_date) between 2 and 3;*

**Find the avg sale of a day**

Example - on 2019-04-03

*select avg(quantity\*saleprice(S.SP\_ID)) as averagesale from salesorders S where S.order\_date ='2019-04-03';*

**Find the avg rent of a day**

*select avg(quantity\*rentprice(S.RP\_ID)) as averagerent from rentorders S where S.order\_date='2019-04-03';*

**Find the returning date of products along with rent to the renter having id =1**

*select order\_id, rentproductname(R1.RP\_ID) as Name, return\_date from rentorders R1 where exists ( select R\_ID from rentproduct Rp where Rp.R\_ID=1 AND R1.RP\_ID = Rp.RP\_ID);*

**Name of the Shipping company of order given by customer 1 of the order on 2019-03-03**

*select Sc.Comp\_name , Sc.Contact from shippingcomp Sc where sc.ship\_id = ( select ship\_id from salesorders s1 where cust\_id=1 and order\_date= '2019-05-03');*

**Delivery person who has received an order made by costumer 1 of on 2019-03-03**

*select Del\_Id, DelPerson\_name from deliveryperson p where p.ship\_id =( select Sc.ship\_id from shippingcomp Sc where sc.ship\_id = ( select ship\_id from salesorders s1 where cust\_id=1 and order\_date= '2019-05-03'));*

### **Procedures and functions call**

**History of customer's orders with given ID**

call Saleorderinvoice(1); for customer with id =1

call rentorderinvoice (9); for customer with id=9

**History of products sold by seller with given ID**

*call GetSellerHistory(2); id=2*

**History of products on rent by renter with given ID**

*call Getrenterhistory(15); id=15*

**Total price of a product by product id sold according to date**

*select SP\_ID, Order\_Date, sum(quantity\*saleprice(SP\_ID)) over( partition by SP\_ID order by order\_date ) as totalsale from salesorders where SP\_ID=6;*

***Total rent of a product by product id sold according to date***

*select RP\_ID, Order\_Date, sum(quantity\*rentprice(RP\_ID)) over( partition by RP\_ID order by order\_date ) as totalrent from rentorders where RP\_ID= 15;*

**Filter selling products and rent products by name**

call filtersaleproducts("double bed");

call filterentproducts("double bed");

**Bonus**

**Register a new seller or renter**

We have considered a scenario where seller or renters send the requests to admin of the site to become a part of this business. Now, admin will look into request table. After consulting with other, if he wants to add the seller or renter he will just call the register procedure with input as request ID. The procedure will look about the details of a seller or renter and add to them to their respective database.

*call register(1); - registers a seller*

*call register(2); - renter*

**Price when an offer is going on**

Sales and offers are quite common on Ecommerce websites.We have that feature too. Whenever a sale month will come our store will automatically show the reduced price according to the offer. For example- product with ID=1 has 500 price on normal days, but if the offer is running on 4th Month, the price would be 60% less i.e. 200.

*call offerpriceselling(15);*

*select offerpricesell(S.SP\_ID) as price from salesorders S where S.order\_date ='2019-02-03'; ## this date has 2 products with ID 1 and 15*

*select saleprice(S.SP\_ID) as price from salesorders S where S.order\_date ='2019-02-03'; ##*

## Given below are the PL/SQL Functions and Procedures which we’ve used.

### **Procedures**

CREATE DEFINER=`root`@`localhost` PROCEDURE `**filterentproducts**`(IN product varchar(40))

BEGIN

select RP\_Id, R\_ID, RP\_name,Category\_ID,RP\_Price from saleproducts where RP\_name= product;

END

CREATE DEFINER=`root`@`localhost` PROCEDURE `**filtersaleproducts**`(IN product varchar(40))

BEGIN

select SP\_Id, S\_ID, SP\_name,Category\_ID,SP\_Price from saleproducts where SP\_name= product;

END

CREATE DEFINER=`root`@`localhost` PROCEDURE **`Getrenterhistory`**(In ID integer)

BEGIN

select Order\_ID,rentorders.RP\_ID as RP\_ID ,rentproduct.Rp\_name, rentorders.quantity as RP\_quantity, rp\_price\*quantity as total\_price, Cust\_ID, Ship\_id, Order\_date ,Shipping\_date, return\_date from rentproduct,rentorders where rentproduct.R\_ID= ID and rentproduct.RP\_ID= rentorders.RP\_ID;

END

CREATE DEFINER=`root`@`localhost` PROCEDURE **`GetSellerhistory`**(In ID integer)

BEGIN

select Order\_ID,salesorders.SP\_ID as SP\_ID ,saleproducts.Sp\_name, salesorders.quantity as SP\_quantity, sp\_price\*quantity as total\_price, Cust\_ID, Ship\_id, Order\_date ,Shipping\_date from saleproducts,salesorders where saleproducts.S\_ID= ID and saleproducts.SP\_ID= salesorders.SP\_ID;

END

CREATE DEFINER=`root`@`localhost` PROCEDURE **`register`**(IN ID integer)

BEGIN

declare nam varchar(50);

declare num bigint default 0;

declare typ varchar(50);

declare cit varchar (50);

declare ad varchar (200);

declare zip bigint;

declare s int default 0;

declare r int default 0;

select Request\_type, Store\_name , Person\_contact,city,address,zipcode into typ,nam,num,cit,ad,zip from request where request.Request\_Id=ID;

set s= get\_id\_seller(); -- get last id

set r = get\_id\_renter();-- get last id

if typ="New Seller"

then insert into seller (S\_ID,S\_name,S\_contact,S\_city,S\_address, S\_zipcode) values ( s+1,nam, num, cit,ad, zip);

end if;

if typ="New Renter"

then insert into renter (R\_ID, R\_name,R\_contact,R\_city,R\_address, R\_zipcode) values (r+1, nam, num, cit,ad, zip);

end if;

END

CREATE DEFINER=`root`@`localhost` PROCEDURE **`rentorderinvoice`**(IN ID integer)

BEGIN

select Order\_ID, Ship\_ID , Cust\_ID ,

sum(rentprice(RP\_ID)\*S1.quantity) as Total,

Order\_date, Shipping\_date, return\_Date

from rentorders s1

where Cust\_ID=ID;

END

CREATE DEFINER=`root`@`localhost` PROCEDURE **`Saleorderinvoice`**(IN ID integer)

BEGIN

select Order\_ID, Ship\_ID , Cust\_ID ,

sum(saleprice(SP\_ID)\*S1.quantity) as Total,

Order\_date, Shipping\_date

from salesorders s1

where Cust\_ID=ID

group by Order\_ID;

END

### **Functions**

CREATE DEFINER=`root`@`localhost` FUNCTION `**get\_id\_renter`**() RETURNS int(11)

READS SQL DATA

DETERMINISTIC

BEGIN

declare q int default 0;

SELECT R\_ID into q FROM renter order BY R\_id DESC LIMIT 1;

RETURN q;

END

CREATE DEFINER=`root`@`localhost` FUNCTION **`get\_id\_seller`**() RETURNS int(11)

READS SQL DATA

DETERMINISTIC

BEGIN

declare q int default 0;

SELECT S\_ID into q FROM seller order BY S\_id DESC LIMIT 1;

RETURN q;

END

END

CREATE DEFINER=`root`@`localhost` FUNCTION **`saleprice`**(ID Int) RETURNS int(11)

READS SQL DATA

DETERMINISTIC

BEGIN

DECLARE q int default 0;

select SP\_Price into q

from SaleProducts

where SP\_ID =ID;

RETURN q;

END

CREATE DEFINER=`root`@`localhost` FUNCTION `**rentprice**`(ID Int) RETURNS int(11)

READS SQL DATA

DETERMINISTIC

BEGIN

DECLARE q int default 0;

select RP\_Price into q

from rentproduct

where RP\_ID =ID;

RETURN q;

END

CREATE DEFINER=`root`@`localhost` FUNCTION **`offerpricesell`**(ID Int) RETURNS int(11)

READS SQL DATA

DETERMINISTIC

BEGIN

DECLARE q int default 0;

##DECLARE f int default 0;

if month(curdate())= 04

then select (S.Sp\_price - S.Sp\_price\*O.off\*0.01) into q from saleproducts S, offerpricesaleprod O where S.SP\_ID=O.SP\_ID AND S.SP\_ID= ID ;

end if;

if month(curdate())<> 04

then select (S.Sp\_price) into q from saleproducts S where S.SP\_ID= ID ;

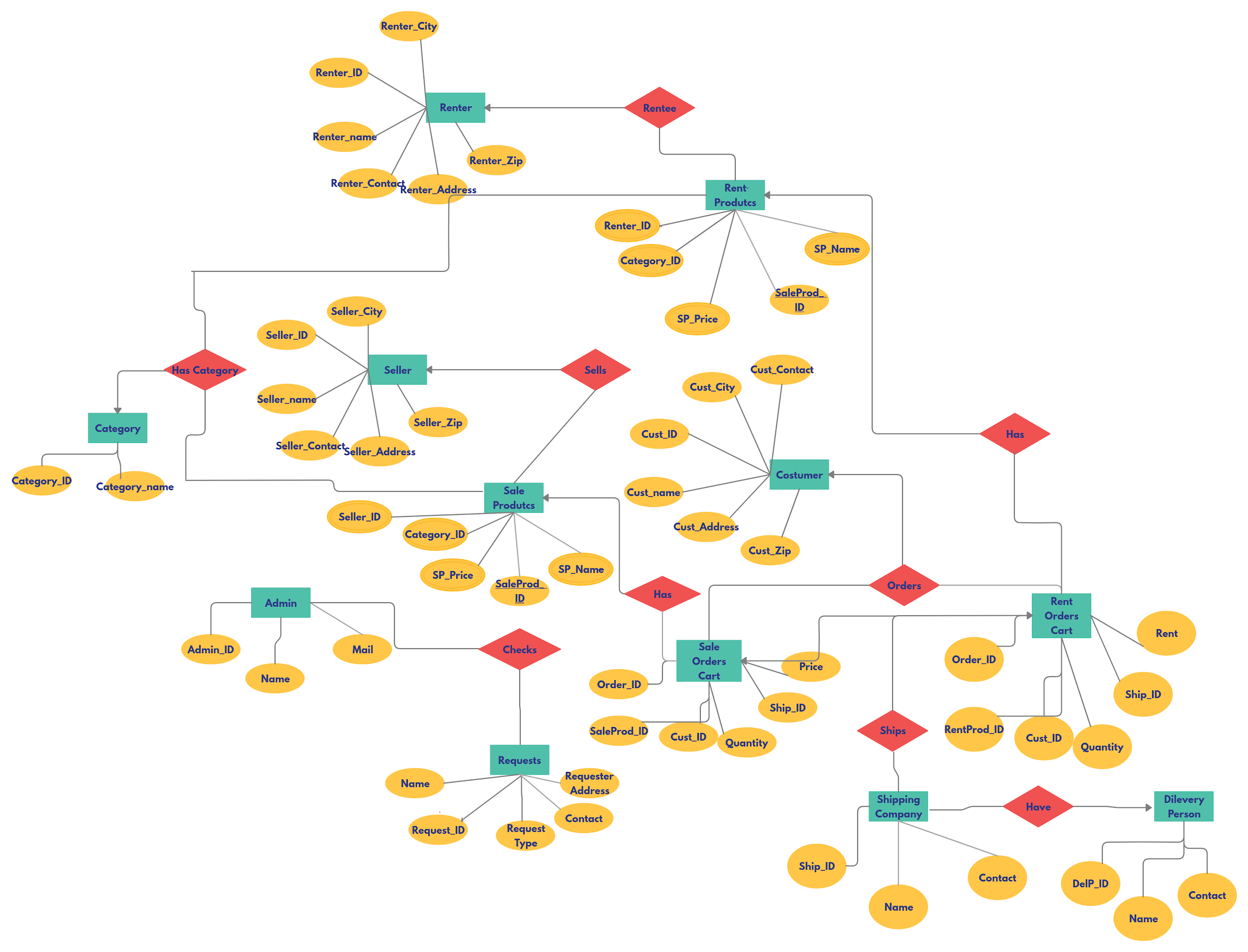
end if;

RETURN q;

END

**E-R Diagram Of The Store**

C



·

·

## Thank You..