

PROJECT REPORT

E-Commerce Grocery System

for
'InstaCart'

In partial fulfilment for the course of

DATABASE MANAGEMENT AND DATABASE DESIGN



NORTHEASTERN UNIVERSITY

Boston, MA

Fall - 2014

-Vaibhav R. Mistry

Table of Contents:

Chapter No		Title	Page No.
1		INTRODUCTION	
	1.1	Abstract	3
	1.2	Purpose of Database	3
	1.3	Project Goals	3
2		USER ROLES AND USE CASE DIAGRAMS	
	2.1	User Roles	4
	2.2	Use Case Diagram	5
3		NORMALIZATION	7
4		BUSINESS RULES	9
5		E-R DIAGRAM	10
6		CONCEPTS IMPLEMENTED	11
7		VIEWS	13
8		SECURITY/USERS	19
9		STORED PROCEDURES	22
10		TRIGGERS	32
11		HIGH LEVEL QUESTIONS	36
12		CONCLUSION	39

1. Introduction

1.1 Abstract:

Groceries are required on a daily basis. As our life becomes busy day by day, it is difficult to remove time for groceries. Here comes in 'Instacart' with their tag line: "never go to a grocery store again". Instacart is the grocery delivery service that delivers the groceries in less than an hour. It connects the customer with personal shoppers in the same area as the customer and delivers the groceries from the local stores.

Customers can choose from a variety of local stores including Whole Foods, Safeway, Costco, Mariano's, and more.

There are two types of Customers: Regular and Express. Express customer fee for the year is \$99. Once the customer is an Express Customer, he can get free deliveries. The regular customer is charged for the shipping.

1.2 Purpose of Database:

As Instacart is an online e-commerce system, it needs to maintain a database for its daily activities like taking orders, shipping them, maintaining the order records and the increase its customer base.

1.3 Project Goals:

To understand the working of the Instacart and understand the concepts of database through their online e-commerce website. The database address the following scenarios:

- 1) Enable customer feedback on the products purchased via the review.
- 2) Enable customers to view their past orders
- 3) Customers can store their payment details so the transactions can occur quickly.
- 4) Maintaining the discounts given to the customers via the discounts and the coupons.
- 5) Each Order consisting of a unique shipping code so that the customers can easily track their order
- 6) Increasing the sales via the data mining techniques implemented .

2. User Roles and Use Case Diagrams

1. User Roles:

A database user has special privileges on tables to insert, modify or delete data in the tables. The user roles supported in this project are:

1) Instacart Database Administrator:

A database administrator is a person who has access to the whole database system. He maintains the transactions for the Instacart. He is considered to be the super user with all the privileges granted to him. With proper approvals he can add new user account, modify the credentials, create reports for the management.

2) Manager:

The Manager has access to the data of all the employees in of Instacart. He has access to manage the salary of the employees. Additional responsibilities include managing product price, discounts and customer coupons.

3) Shipping Employee

The shipping employee is the one who is responsible for shipping the orders to the customers. He has access to all the orders of Instacart. He can modify the shipping status of the order and also can track the order.

4) Financial Accountant

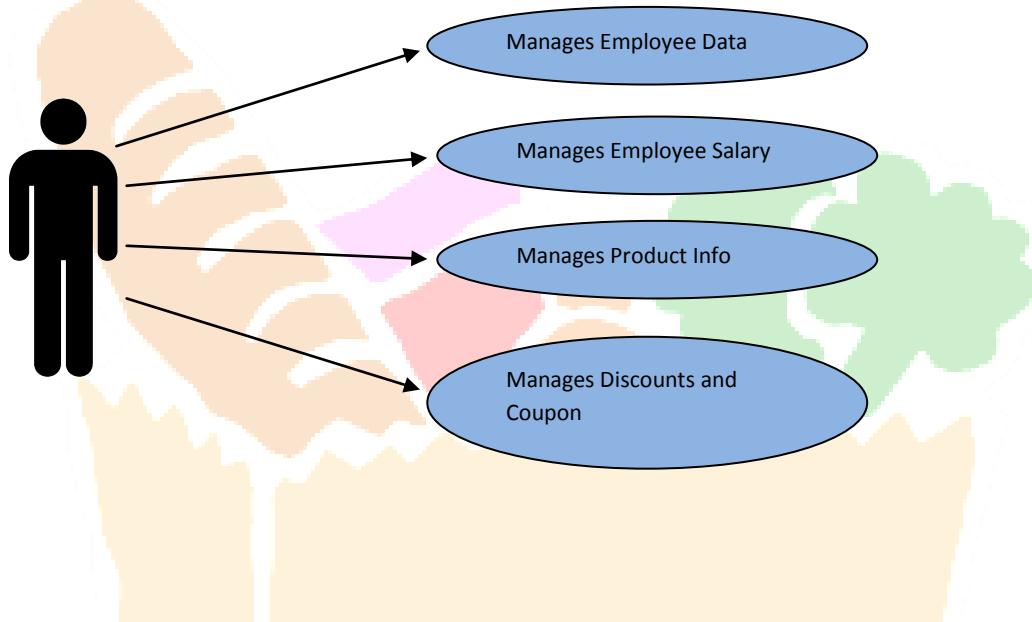
The Financial accountant has access to the financial records of the database. He creates the financial reports and monitors the overall performance of Instacart.

5) Customer

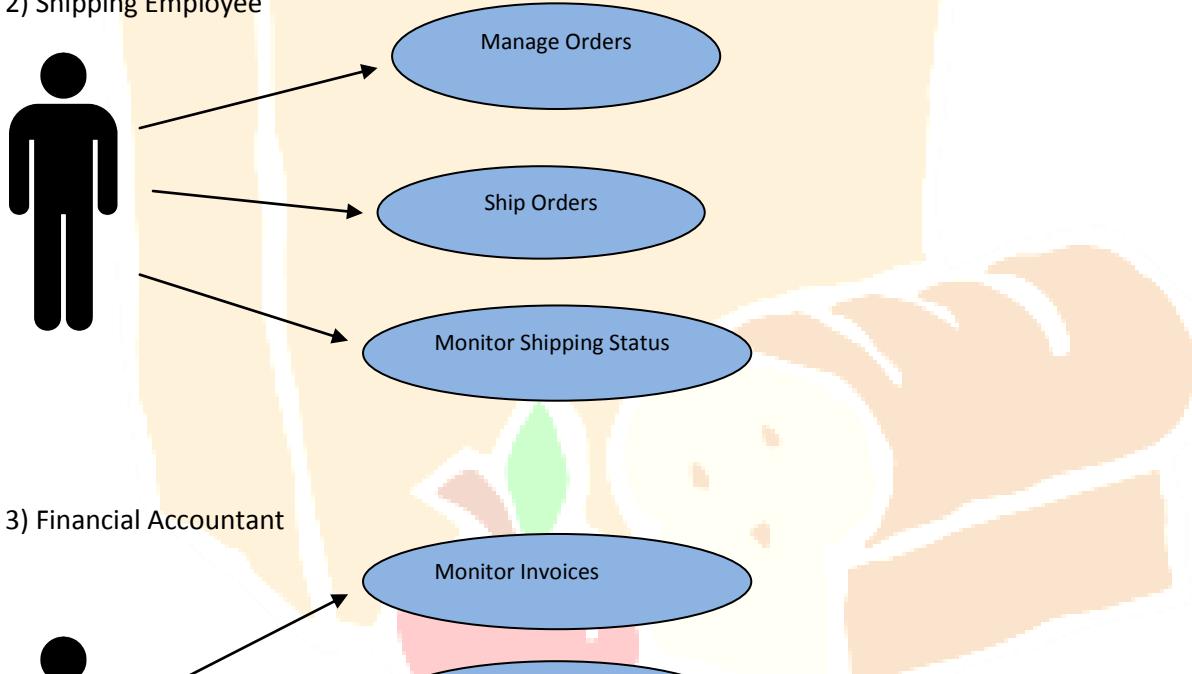
The whole and sole of the Enterprise. Customer is the one who orders products and helps in generating the revenue for Instacart. There are two types of customer- Regular and Express. By default the customer is a regular customer unless he wishes to pay a premium amount to become an express customer. The benefits of being an express customer is that the shipping costs are removed on every order. Customer is not only the consumer of the data but also the producer of the data. He can provide a feedback of the products via the ratings.

2. Use- Case Diagrams:

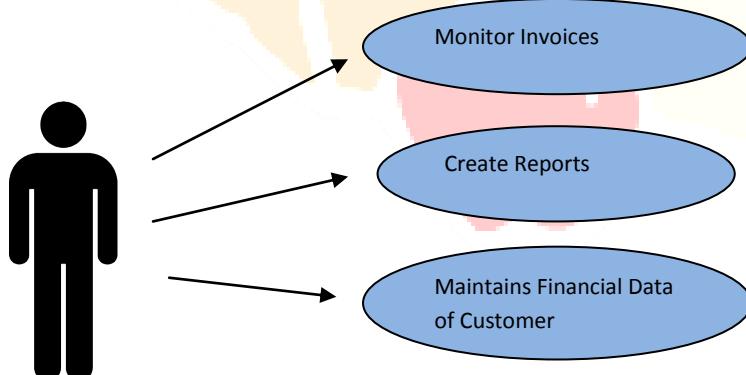
1) Manager:



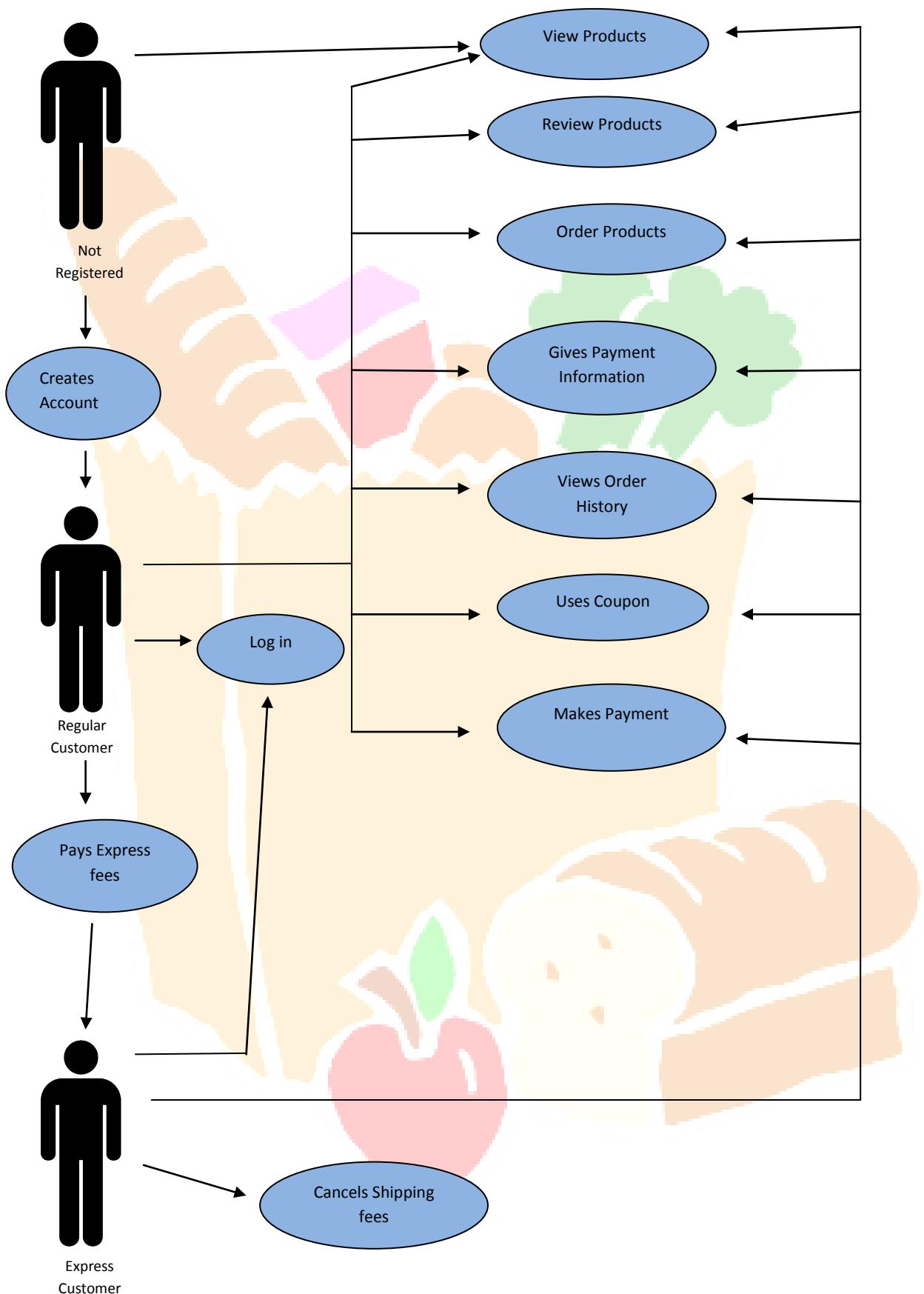
2) Shipping Employee



3) Financial Accountant



4) Customer:



3. Normalization

Normalization is the process of decomposing relations with anomalies to produce smaller, well-structured relations.

Normalization are of types: 1NF, 2NF, 3NF, Boyce/Codd NF, 4NF, 5NF, Domain Key Normal Form. Of which 1NF,2NF,3NF and Boyce Codd depend on Functional dependencies.

1st Normal form:

For tables to be in 1st NF there should not be any multi-valued attributes. Each column should be atomic.

2nd Normal Form

For tables to be in 2NF, they should be in 1NF and partial dependencies should be removed. It means that all non-key fields should depend on the key field.

3rd Normal Form

For tables to be in 3NF, they should be in 2NF and all the transitive dependencies should be removed.

Boyce/Codd Normal Form:

For Tables to be in BCNF, they should be in 3NF and every determinant should be a primary key.

4th Normal Form

For 4NF, the tables should be in BCNF and it should not contain any multi-valued dependencies.

5th Normal Form

For 5NF, the tables should be in 4NF and the tables shouldn't contain any related multi-valued dependencies.

Domain Key Normal Form.

For table to be in DKNF, the table should consist no constraints except domain constraints and key constraints.

Following Normalization steps were carried out in the project:

- 1) For my project during the initial phase the tables such as the person consisted of an attribute address which had multiple values such as the street name, city, state and zip. To convert the table in 1st NF, the data was separated. Likewise many other multivalued attributes were removed from the tables.
- 2) All the tables have a single primary key attribute which has been kept as Auto-Increment. So partial dependencies have been eliminated. This is to comply with 2NF
- 3) Person's table initially included information such as the email, phone which could determine the person's name so separate tables were made for them.
- 4) Tax percentage and state were initially included in the Orders table. But the state could determine the tax percentage which violated the 3NF, so they were separated out.
- 5) The product table consisted of department and department id but this violated 3NF as through department id, department could be found out hence they were separated into another table.
- 6) For customer, the payment details were first included in the customer table. This created transitive dependencies that using card details, other payment details could be found out, so they were separated into another table.
- 7) Invoice details were separated from the orders table to a new table, to comply with 3NF so that transitive dependencies could be removed.

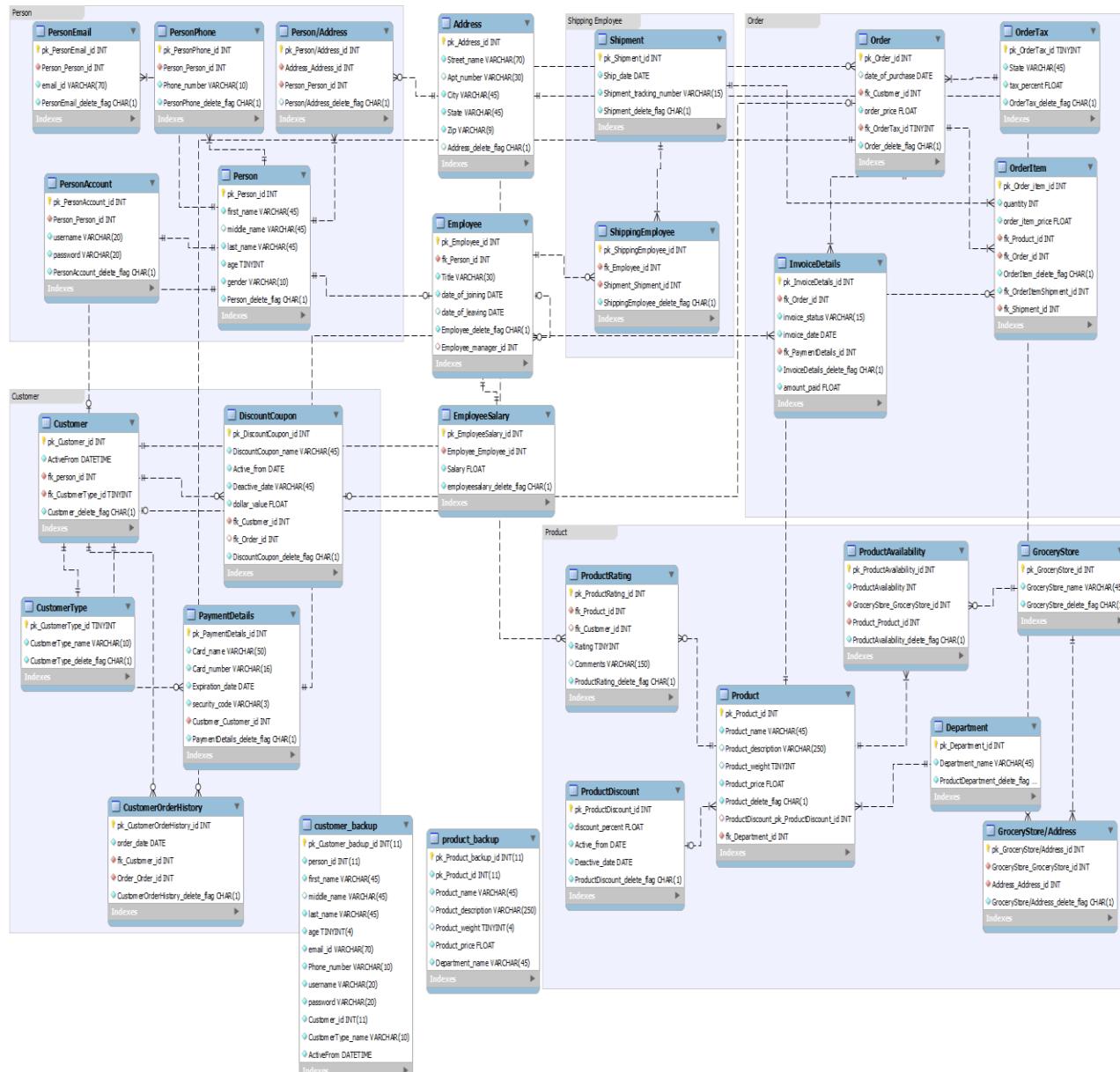
Thus all the tables were converted into 3NF, since all the tables consist of primary key, they are in BCNF as the determinant is the primary key.

4. Business Rules

The following business rules were implemented while designing the Instacart database:

- 1) The data is persisted in the database. None of the data is deleted.
- 2) The person who can access the database is either an employee of Instacart or a customer.
- 3) Customers are of two types: Regular and Express. For express customers, the shipping costs are removed.
- 4) A person must have an email, phone and address associated with him.
- 5) A Customer may have none or more than one payment methods. But to create an order, he requires the payment details.
- 6) A Customer has to be of one of the type: Regular or Express.
- 7) A customer can have none or more than one discount coupon. But a Discount coupon needs to have a customer.
- 8) A product can have none or more than one rating. But a rating needs to have a customer associated.
- 9) Every product belongs to a department and a department should at least have a product.
- 10) A product can have none or more than one discount but a discount needs to be associated with a product
- 11) A product can be associated with one or more grocery store.
- 12) The availability depends on the grocery store.
- 13) A Shipment employee can handle more than one shipments
- 14) A customer can place none or more than one order but the order needs to have a customer associated with him
- 15) An Order must have a tax associated with it whereas a tax can be applied on multiple orders.
- 16) Every order needs to have card details.
- 17) Every invoice has customer payment details

5. E-R Diagram



6. Concepts Used

1) Primary Key:

A primary key is an attribute or combination of attributes which helps in uniquely identifying the row.

The value of primary key cannot be null.

The value should be unique.

Normally we use auto-increment for the fields so that values are not repeated.

All the primary key attributes in the project are prefixed by "pk_"

2) Foreign Key

A foreign key is the attribute which maintains the referential integrity and points to the primary key where it is referenced.

The value of foreign key can be null and be repetitive.

Foreign keys can be of following types:

1 to 1 relationship: Here the foreign key is kept on the table with minimum nulls.

1 to N relationship: Here foreign key is kept on the many side.

M to N relationship : Here we establish a new link table with primary keys of both the tables as foreign keys.

In the project the foreign keys are denoted with "fk_" suffix.

3) Relationships:

a) Unary Relationship: Here the relationship exist between the same entity.

In the project Unary relationship is created in the Employee Table where there Employee who is the manager can manage multiple staff under him. Hence a one to many relationship is created in the Employee table.

b) Binary Relationship: Here the relationship exists between two different entities. This is the most common type of relationship. There can be 1:1, 1:N or M:N type of relationship.

In the project many binary relationships are established.

example:

Employee table is linked with Salary having 1:1 relationship

Order linked with OrderItem table having 1:N relationship

Person and Address having M:N relationship so a link table was created Person/Address .

c) Ternary Relationship: Here the relationship exists between three different entities.

In the project no such relationship exists.

4) Indexes:

Indexes help in getting the search results quicker. When Joins are performed on the tables the indexing can help in getting faster results. Indexes are of following types:

- a) Primary key Index
- b) Foreign key Index
- c) Unique key Index
- d) Index
- e) Full- Text Index.

In the project following indexes were defined:

Table Name	Attribute	Index Type
Person	last_name	UNIQUE
PersonEmail	email_id	FULLTEXT
CustomerType	customerType_name	INDEX
Product	product_price	INDEX
InvoiceDetails	invoice_status	INDEX
PaymentDetails	card_number	INDEX

SQL Code:

```
CREATE UNIQUE INDEX last_name_idx ON Person (last_name);
CREATE FULLTEXT INDEX email_idx ON PersonEmail (email_id);
CREATE INDEX customer_type_idx ON CustomerType (customerType_name);
CREATE INDEX product_price_idx ON Product (product_price);
CREATE INDEX invoice_status_idx ON InvoiceDetails (invoice_status);
CREATE INDEX card_number_idx ON PaymentDetails (card_number);
```

7. Views

Views act as virtual tables . They are stored queries which when invoked produces a result set. Users can be assigned to these views. Any changes made in the data in the views also affect the base tables which the view is pointing to.

In this project as the data needs to be persistent, there are two views made of each table. One with delete flag as 'F' and the other will be the complete view of the table. This is done so that when a delete action needs to be performed, the data administrator would simply change the delete_flag = 'T', Hence to the user the data would appear as deleted but in the back the data would persist.

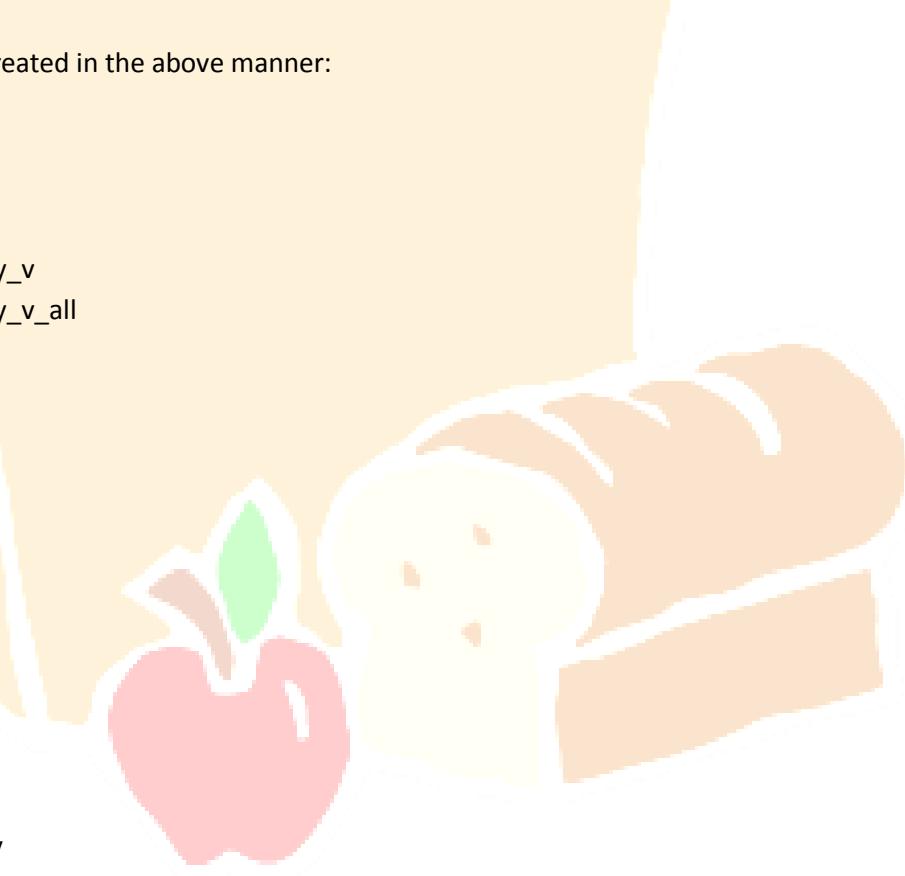
The views with suffix "_v" have the data with delete_flag ='F'. The views with complete view of table have suffix "_v_all".

One such example is shown below:

```
CREATE VIEW customer_v AS
SELECT pk_Customer_id, ActiveFrom , fk_person_id , fk_CustomerType_id  FROM customer where
Customer_delete_flag='F';
```

pk_Customer_id	ActiveFrom	fk_person_id	fk_CustomerType_id
1	2014-01-10 01:02:00	1	1
2	2012-08-08 03:02:00	2	2
3	2011-03-20 01:02:00	3	1
4	2014-12-02 14:00:35	8	2

```
CREATE VIEW customer_v_all AS
SELECT * FROM customer;
```



A screenshot of MySQL Workbench showing the creation of a view named 'customer_v_all'. The interface includes a left sidebar with navigation options like 'MANAGEMENT' and 'SCHEMAS', and a central workspace displaying the SQL query and its results.

SQL Editor:

```
customer_view_3      shipping_employee_view      shipping_employee_view_2      Procedures      SQL File 43*      Triggers      Users      SQLQueries*      Administration - Data Export      customer_v      customer_v_all
```

Result Set:

pk_Customer_id	ActiveFrom	fk_person_id	fk_CustomerType_id	Customer_delete_flag
1	2014-01-10 01:02:00	1	1	F
2	2012-08-08 03:02:00	2	2	F
3	2011-03-20 01:02:00	3	1	F
4	2014-12-02 14:00:35	8	2	F

Information Panel:

View: customer_v_all

Columns:

- pk_Customer_id int(11)
- ActiveFrom datetime
- fk_person_id int(11)
- fk_CustomerType_id tinyint(4)
- Customer_delete_flag char(1)

Object Info: Session

Output Panel:

customer_v_all 1 > Read Only

Action Output

399 22:51:05 SELECT * FROM instacart_db.customer_v LIMIT 0, 1000
4 row(s) returned
Duration / Fetch: 0.000 sec / 0.000 sec

400 22:52:09 SELECT * FROM instacart_db.customer_v_all LIMIT 0, 1000
4 row(s) returned
Duration / Fetch: 0.000 sec / 0.000 sec

Query Completed

Following views were created in the above manner:

address_v
 address_v_all
 customer_v
 customer_v_all
 customer_order_history_v
 customer_order_history_v_all
 customer_type_v
 customer_type_v_all
 department_v
 department_v_all
 discount_coupon
 discount_coupon_all
 employee_v
 employee_v_all
 employee_salary_v
 employee_salary_v_all
 grocery_store_v
 grocery_store_v_all
 grocerystore_address_v
 grocerystore_address_v_all
 invoice_details_v
 invoice_details_v_all
 order_v
 order_v_all

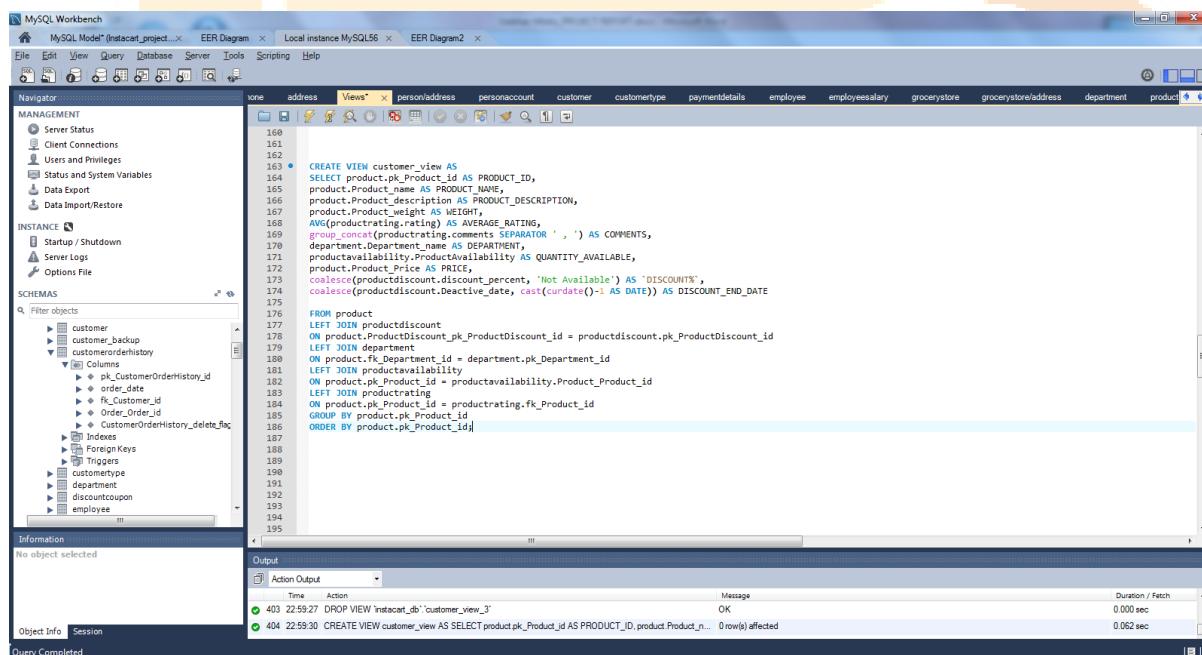
order_item_v
order_item_v_all
order_tax_v
order_tax_v_all
payment_details_v
payment_details_v_all
person_v
person_v_all
person_address_v
person_address_v_all
person_account_v
person_account_v_all
person_email_v
person_email_v_all
person_phone_v
person_phone_v_all
product_v
product_v_all
product_availability_v
product_availability_v_all
product_discount_v
product_discount_v_all
product_rating_v
product_rating_v_all
shipment_v
shipment_v_all
shipping_employee_v
shipping_employee_v_all

DBMS Project - Fall 2014

- 1) The following view was created for customer so that he can see the complete product details before making a purchase

```
CREATE VIEW customer_view AS
SELECT product.pk_Product_id AS PRODUCT_ID,
product.Product_name AS PRODUCT_NAME,
product.Product_description AS PRODUCT_DESCRIPTION,
product.Product_weight AS WEIGHT,
AVG(productrating.rating) AS AVERAGE_RATING,
group_concat(productrating.comments SEPARATOR ', ') AS COMMENTS,
department.Department_name AS DEPARTMENT,
productavailability.ProductAvailability AS QUANTITY_AVAILABLE,
product.Product_Price AS PRICE,
coalesce(productdiscount.discount_percent, 'Not Available') AS 'DISCOUNT%',
coalesce(productdiscount.Deactive_date, cast(curdate()-1 AS DATE)) AS DISCOUNT_END_DATE

FROM product
LEFT JOIN productdiscount
ON product.ProductDiscount_pk_ProductDiscount_id = productdiscount.pk_ProductDiscount_id
LEFT JOIN department
ON product.fk_Department_id = department.pk_Department_id
LEFT JOIN productavailability
ON product.pk_Product_id = productavailability.Product_Product_id
LEFT JOIN productrating
ON product.pk_Product_id = productrating.fk_Product_id
GROUP BY product.pk_Product_id
ORDER BY product.pk_Product_id;
```



DBMS Project - Fall 2014

The output is as shown below:

The screenshot shows the MySQL Workbench interface. In the top navigation bar, tabs include 'MySQL Model', 'Instacart_project...x', 'EER Diagram', 'Local instance MySQL56', and 'EER Diagram2'. Below the tabs, the main area displays a query editor with the following SQL code:

```
1 • SELECT * FROM instacart_db.customer_view;
```

The results pane shows a table with the following data:

PRODUCT_ID	PRODUCT_NAME	PRODUCT_DESCRIPTION	WEIGHT	AVERAGE_RATING	COMMENTS	DEPARTMENT	QUANTITY_AVAILABLE	PRICE	DISCOUNT%	DISCOUNT_END_DATE
1	Bananas	Box of Bananas	3	5.0000	Good Purchase , Good Purchase , Not worth , Not worth	Produce	50	1.69	Not Available	2014-12-01
2	Corn	Nature fresh corn	8	7.0000	Good Purchase , Good Purchase	Produce	30	6.09	20	2015-01-01
3	Milk	Kirkland milk	1	8.0000	Good Purchase , Good Purchase	Dairy	90	3.09	Not Available	2014-12-01
4	Butter	Stops yummy butter	4	9.0000	Good Purchase , Good Purchase	Dairy	66	9.79	20	2015-01-01
5	Chiken	Fresh curvy cut	7	10.0000	Worth the Purchase , Worth the Purchase	Meat/SeaFood	20	15.33	Not Available	2014-12-01
6	Fish	Tilapia fish	3	11.0000		Meat/SeaFood	90	17.91	20	2015-01-01
7	mushrooms	Organic Mushrooms	1	12.0000		Produce	10	3.21	Not Available	2014-12-01

Below the results, the 'Information' pane shows the definition of the 'customer_view' view:

```
VIEW: customer_view
Columns:
PRODUCT_ID
PRODUCT_NAME
PRODUCT_DESCRIPTION
WEIGHT
AVERAGE_RATING
COMMENTS
DEPARTMENT
QUANTITY_AVAILABLE
PRICE
DISCOUNT%
DISCOUNT_END_DATE
```

The 'Output' pane shows the following log entries:

- 404 22:59:30 CREATE VIEW customer_view AS SELECT product_pk_Product_id AS PRODUCT_ID, product.Product_name AS PRODUCT_NAME, ... 0 rows affected Duration / Fetch 0.052 sec
- 405 23:02:09 SELECT * FROM instacart_db.customer_view LIMIT 0, 1000 7 row(s) returned 0.015 sec / 0.000 sec

2) Another view is created for the shipping employee so that he can view all the shipping details :

```
CREATE VIEW shipping_employee_view AS
SELECT
shipment.pk_Shipment_id AS SHIPMENT_ID,
shipment.Shipment_tracking_number AS TRACKING_NUMBER,
orderitem.quantity AS QUANTITY,
orderitem.fk_Product_id AS PRODUCT_ID,
product.Product_name AS PRODUCT_NAME,
`order`.pk_Order_id AS ORDER_ID,
`order`.date_of_purchase AS ORDER_DATE,
customer.pk_Customer_id AS CUSTOMER_ID,
concat(person.first_name, ' ', coalesce(person.middle_name, ' ') , person.last_name) AS CUSTOMER_NAME,
personemail.email_id AS EMAIL_ID,
concat(address.Street_name, ' ', address.Apt_number, ' ', address.City, ' ', address.state, ' ', address.zip) AS SHIP_ADDRESS
FROM `order`
INNER JOIN customer
ON `order`.fk_Customer_id = customer.pk_Customer_id
INNER JOIN person
ON person.pk_Person_id = customer.fk_person_id
INNER JOIN `person/address`
ON `person/address`.Person_Person_id = person.pk_Person_id
INNER JOIN address
ON address.pk_Address_id = `person/address`.Address_Address_id
```

```

INNER JOIN personemail
ON person.pk_Person_id = personemail.Person_Person_id
INNER JOIN orderitem
ON orderitem.fk_Order_id = order.pk_Order_id
INNER JOIN shipment
ON shipment.pk_Shipment_id = orderitem.fk_Shipment_id
INNER JOIN product
ON product.pk_Product_id = orderitem.fk_Product_id
GROUP BY product.pk_Product_id
ORDER BY shipment.pk_Shipment_id;

```

The output is as shown below:

SHIPMENT_ID	TRACKING_NUMBER	QUANTITY	PRODUCT_ID	PRODUCT_NAME	ORDER_ID	ORDER_DATE	CUSTOMER_ID	CUSTOMER_NAME	EMAIL_ID	SHIP_ADDRESS
1	652145781	1	3	Milk	1	2014-11-11	1	Vahav RMistry	vahav@gmail.com	181 Hemenway st., Boston, MA, 2115
1	652145781	2	1	Bananas	1	2014-11-11	1	Vahav RMistry	vahav@gmail.com	181 Hemenway st., Boston, MA, 2115
1	652145781	1	5	Chicken	1	2014-11-11	1	Vahav RMistry	vahav@gmail.com	181 Hemenway st., Boston, MA, 2115
2	5147541207	1	6	Fajita	2	2014-10-01	2	Richa PSharma	richa@hotmail.com	132 Pkeming st., San Jose, CA, 95103
4	5123657810	1	4	Butter	3	2014-02-22	3	Ankit EGala	ankit@gmail.com	165 Hemenway st., Boston, MA, 2115

8. Security/ Users

Database security should be considered of the prime importance when a database is created.

Following factors need to be considered when security is implemented in the database:

- a) Select good passwords
- b) Restricting the users by limiting the access given to them
- c) Providing physical security

In the project following privileges are given to the users depending on their roles on the views created above.

1) Instacart Database Administrator

```
CREATE USER 'instacart_db_admin'@'%';
GRANT ALL PRIVILEGES ON instacart_db.* TO 'instacart_db_admin'@'%';
```

2) Manager

```
CREATE USER 'manager'@'%';
GRANT SELECT,UPDATE,INSERT ON instacart_db.employee_v TO 'manager'@'%';
GRANT SELECT,UPDATE,INSERT ON instacart_db.employeesalary_v TO 'manager'@'%';
GRANT SELECT,UPDATE,INSERT ON instacart_db.product_v TO 'manager'@'%';
GRANT SELECT,UPDATE,INSERT ON instacart_db.product_availability_v TO 'manager'@'%';
GRANT SELECT,UPDATE,INSERT ON instacart_db.product_discount_v TO 'manager'@'%';
GRANT SELECT,UPDATE,INSERT ON instacart_db.grocery_store_v TO 'manager'@'%';
GRANT SELECT,UPDATE,INSERT ON instacart_db.grocerystore_address_v TO 'manager'@'%';
GRANT SELECT,UPDATE,INSERT ON instacart_db.department_v TO 'manager'@'%';
GRANT SELECT,UPDATE,INSERT ON instacart_db.shipping_employee_v TO 'manager'@'%';
GRANT SELECT ON instacart_db.product_rating_v TO 'manager'@'%';
GRANT SELECT,UPDATE,INSERT ON instacart_db.order_v TO 'manager'@'%';
GRANT SELECT ON instacart_db.order_item_v TO 'manager'@'%';
GRANT SELECT ON instacart_db.order_tax_v TO 'manager'@'%';
GRANT SELECT ON instacart_db.invoice_details_v TO 'manager'@'%';
GRANT SELECT ON instacart_db.shipment_v TO 'manager'@'%';
GRANT SELECT,UPDATE,INSERT ON instacart_db.address_v TO 'manager'@'%';
GRANT SELECT,UPDATE,INSERT ON instacart_db.person_v TO 'manager'@'%';
GRANT SELECT,UPDATE,INSERT ON instacart_db.person_address_v TO 'manager'@'%';
GRANT SELECT,UPDATE,INSERT ON instacart_db.person_email_v TO 'manager'@'%';
GRANT SELECT,UPDATE,INSERT ON instacart_db.person_phone_v TO 'manager'@'%';
GRANT SELECT,UPDATE,INSERT ON instacart_db.discount_coupon TO 'manager'@'%';
GRANT SELECT ON instacart_db.customer_v TO 'manager'@'%';
```

3) Shipping Employee

```
CREATE USER 'shipping_employee'@'%';
GRANT SELECT,UPDATE,INSERT ON instacart_db.shipment_v TO 'shipping_employee'@'%';
GRANT SELECT ON instacart_db.order_v TO 'shipping_employee'@'%';
GRANT SELECT ON instacart_db.order_item_v TO 'shipping_employee'@'%';
GRANT SELECT ON instacart_db.order_v TO 'shipping_employee'@'%';
GRANT SELECT ON instacart_db.invoice_details_v TO 'shipping_employee'@'%';
GRANT SELECT ON instacart_db.order_v TO 'shipping_employee'@'%';
GRANT SELECT,UPDATE,INSERT ON instacart_db.address_v TO 'shipping_employee'@'%';
GRANT SELECT,UPDATE,INSERT ON instacart_db.person TO 'shipping_employee'@'%';
GRANT SELECT,UPDATE,INSERT ON instacart_db.person_email_v TO 'shipping_employee'@'%';
GRANT SELECT,UPDATE,INSERT ON instacart_db.person_phone_v TO 'shipping_employee'@'%';
GRANT SELECT,UPDATE,INSERT ON instacart_db.person_address_v TO 'shipping_employee'@'%';
GRANT SELECT ON instacart_db.product_v TO 'shipping_employee'@'%';
GRANT SELECT ON instacart_db.product_availability_v TO 'shipping_employee'@'%';
GRANT SELECT ON instacart_db.grocery_store_v TO 'shipping_employee'@'%';
```

4) Financial Accountant:

```
CREATE USER 'financial_accountant'@'%';
GRANT SELECT,UPDATE,INSERT ON instacart_db.order_tax_v TO 'financial_accountant'@'%';
GRANT SELECT,UPDATE,INSERT ON instacart_db.invoice_details_v TO 'financial_accountant'@'%';
GRANT SELECT,UPDATE,INSERT ON instacart_db.order_v TO 'financial_accountant'@'%';
GRANT SELECT,UPDATE,INSERT ON instacart_db.order_item_v TO 'financial_accountant'@'%';
GRANT SELECT,UPDATE,INSERT ON instacart_db.order_tax_v TO 'financial_accountant'@'%';
GRANT SELECT ON instacart_db.discount_coupon TO 'financial_accountant'@'%';
GRANT SELECT ON instacart_db.product_discount_v TO 'financial_accountant'@'%';
GRANT SELECT ON instacart_db.order_tax_v TO 'financial_accountant'@'%';
GRANT SELECT,UPDATE,INSERT ON instacart_db.person TO 'financial_accountant'@'%';
GRANT SELECT,UPDATE,INSERT ON instacart_db.person_email_v TO 'financial_accountant'@'%';
GRANT SELECT,UPDATE,INSERT ON instacart_db.person_phone_v TO 'financial_accountant'@'%';
GRANT SELECT,UPDATE,INSERT ON instacart_db.person_address_v TO 'financial_accountant'@'%';
```

5) Customer:

```
CREATE USER 'customer'@'%';
GRANT SELECT,UPDATE,INSERT ON instacart_db.customer_v TO 'customer'@'%';
GRANT SELECT,UPDATE,INSERT ON instacart_db.payment_details_v TO 'customer'@'%';
GRANT SELECT ON instacart_db.customer_type_v TO 'customer'@'%';
GRANT SELECT ON instacart_db.discount_coupon TO 'customer'@'%';
GRANT SELECT ON instacart_db.customer_order_history_v TO 'customer'@'%';
GRANT SELECT,UPDATE,INSERT ON instacart_db.product_rating_v TO 'customer'@'%';
GRANT SELECT ON instacart_db.product_v TO 'customer'@'%';
GRANT SELECT ON instacart_db.product_availability_v TO 'customer'@'%';
GRANT SELECT ON instacart_db.grocery_store_v TO 'customer'@'%';
GRANT SELECT ON instacart_db.product_v TO 'customer'@'%';
GRANT SELECT, INSERT ON instacart_db.order_item_v TO 'customer'@'%';
GRANT SELECT, INSERT ON instacart_db.order_v TO 'customer'@'%';
GRANT SELECT ON instacart_db.department TO 'customer'@'%';
GRANT SELECT,INSERT ON instacart_db.invoice_details_v TO 'customer'@'%';
GRANT SELECT,UPDATE,INSERT ON instacart_db.person TO 'customer'@'%';
GRANT SELECT,UPDATE,INSERT ON instacart_db.person_email_v TO 'customer'@'%';
GRANT SELECT,UPDATE,INSERT ON instacart_db.person_phone_v TO 'customer'@'%';
GRANT SELECT,UPDATE,INSERT ON instacart_db.person_address_v TO 'customer'@'%';
```

9. Stored Procedures

A) To enter the product details:

DELIMITER !!!

```
CREATE PROCEDURE insert_product_data (product_name varchar(45), product_description  
varchar(250), weight tinyint(4), price float, department varchar(45),  
quantity int(10), grocery_store varchar(45))  
  
BEGIN  
IF EXISTS (SELECT pk_GroceryStore_id FROM grocerystore WHERE GroceryStore_name =  
grocery_store)  
THEN  
SET @grocery_s_id = (SELECT pk_GroceryStore_id FROM grocerystore WHERE GroceryStore_name =  
grocery_store);  
ELSE  
INSERT INTO grocerystore values (null, GroceryStore_name, 'F');  
SET @grocery_s_id = (SELECT MAX(pk_GroceryStore_id) FROM grocerystore);  
END IF;  
IF EXISTS (SELECT pk_Department_id FROM department WHERE Department_name=department)  
THEN  
SET @dept_id =(SELECT pk_Department_id FROM department WHERE  
Department_name=department);  
ELSE  
INSERT INTO department values (null,department,'F');  
SET @dept_id = (SELECT MAX(pk_Department_id) FROM department);  
END IF;  
INSERT INTO product values (null, product_name, product_description, weight, price, 'F', null,  
@dept_id);  
SET @prod_id = (SELECT MAX(pk_Product_id) FROM product);  
INSERT INTO productavailability values(null,quantity, @grocery_s_id,@prod_id,'F');  
END; !!!
```

DBMS Project - Fall 2014

The screenshot shows the MySQL Workbench interface with the 'Procedures' tab selected. A stored procedure named 'insert_product_data' is being created. The code defines variables for product name, description, weight, price, and department, and then inserts them into the 'product' and 'productavailability' tables.

```

DELIMITER !!!
CREATE PROCEDURE insert_product_data (product_name varchar(45), product_description varchar(250), weight tinyint(), price float, department varchar(45),
                                     quantity int(10), grocery_store varchar(45))
BEGIN
    IF EXISTS (SELECT pk_GroceryStore_id FROM grocerystore WHERE GroceryStore_name = grocery_store)
    THEN
        SET @grocery_s_id = (SELECT pk_GroceryStore_id FROM grocerystore WHERE GroceryStore_name = grocery_store);
        INSERT INTO grocerystore values (null, GroceryStore_name, 'F');
        SET @grocery_s_id = (SELECT MAX(pk_GroceryStore_id) FROM grocerystore);
    ELSE
        INSERT INTO departments values (null,department,'F');
        SET @dept_id = (SELECT MAX(pk_Department_id) FROM department);
        INSERT INTO product values (null,product_name,product_description,weight,price, null, @dept_id);
        SET @prod_id = (SELECT MAX(pk_Product_id) FROM product);
        INSERT INTO productavailability values(null,quantity,@grocery_s_id,@prod_id,'F');
    END IF;
END !!!

```

CALL insert_product_data ('mushrooms', 'Organic Mushrooms', 1, 3.21, 'produce', 10, 'Whole Foods');

The screenshot shows the MySQL Workbench interface with the 'SQL File 43*' tab selected. The stored procedure 'insert_product_data' is called with the specified parameters. The log output shows the execution details, including the creation of a new grocery store entry and the insertion of the product and its availability.

```

1 CALL insert_product_data ('mushrooms', 'Organic Mushrooms', 1, 3.21, 'produce', 10, 'Whole Foods')
2

```

Time	Action	Message	Duration / Fetch
408 23:04:18	CREATE VIEW shipping_employee_view AS SELECT shipment_pk_Shipment_ID AS SHIPMENT_ID, shipment_Shipment...	0 row(s) affected	0.016 sec
409 23:09:53	SELECT * FROM instacart_db_shipping_employee_view LIMIT 0,1000	5 row(s) returned	0.140 sec / 0.000 sec

The updated tables are shown below:

DBMS Project - Fall 2014

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

customer_view_2 customer_view_3 customer_view_2 customer_view_4 customer_view_3 shipping_employee_view shipping_employee_view_2 Procedures* SQL File 43* SQL File 44*

Result Set Filter:

pk_Product_id	Product_name	Product_description	Product_weight	Product_price	Product_delete_flag	ProductDiscount_pk_ProductDiscount_id	fk_Department_id
1	Bananas	Box of Bananas	3	1.63	F	NULL	1
2	Com	Nature fresh corn	8	6.09	F	1	1
3	Milk	Kirkland milk	1	3.09	F	NULL	2
4	Butter	Stops yummy butter	4	9.79	F	1	2
5	Chicken	Fresh curvy cut	7	15.33	F	NULL	3
6	Fish	Tilapia fish	3	17.91	F	1	3
7	mushrooms	Organic Mushrooms	1	3.21	F	NULL	1

Output

Action Output

Time	Action	Message	Duration / Fetch
352 14:15:12	select * from employee LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec
353 14:15:43	select * from address LIMIT 0, 1000	11 row(s) returned	0.000 sec / 0.000 sec
354 14:18:22	select * from grocerystore LIMIT 0, 1000	3 row(s) returned	0.031 sec / 0.000 sec
355 14:18:49	CALL insert_product_data ('mushrooms', 'Organic Mushrooms', 1, 3.21, 'produce', 10, 'Whole Foods')	1 row(s) affected	0.094 sec
356 14:18:56	select * from grocerystore LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec
357 14:19:10	select * from product LIMIT 0, 1000	7 row(s) returned	0.000 sec / 0.000 sec

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

customer_view_2 customer_view_3 customer_view_2 customer_view_4 customer_view_3 shipping_employee_view shipping_employee_view_2 Procedures* SQL File 43* SQL File 44*

Result Set Filter:

pk_ProductAvailability_id	ProductAvailability	GroceryStore_GroceryStore_id	Product_Product_id	ProductAvailability_delete_flag
5	90	1	3	F
6	50	2	3	F
7	66	2	4	F
8	70	3	4	F
9	20	2	5	F
10	30	3	5	F
11	5	3	6	F
12	90	1	6	F
13	10	1	7	F

Output

Action Output

Time	Action	Message	Duration / Fetch
353 14:15:43	select * from address LIMIT 0, 1000	11 row(s) returned	0.000 sec / 0.000 sec
354 14:18:22	select * from grocerystore LIMIT 0, 1000	3 row(s) returned	0.031 sec / 0.000 sec
355 14:18:49	CALL insert_product_data ('mushrooms', 'Organic Mushrooms', 1, 3.21, 'produce', 10, 'Whole Foods')	1 row(s) affected	0.094 sec
356 14:18:56	select * from grocerystore LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec
357 14:19:10	select * from product LIMIT 0, 1000	7 row(s) returned	0.000 sec / 0.000 sec
358 14:21:13	select * from productavailability LIMIT 0, 1000	13 row(s) returned	0.016 sec / 0.000 sec

DBMS Project - Fall 2014

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator

MANAGEMENT

- Server Status
- Client Connections
- Users and Privileges
- Status and System Variables
- Data Export
- Data Import/Restore

INSTANCE

- Startup / Shutdown
- Server Logs
- Options File

SCHEMAS

- Filter objects
- instacart_db

 - Tables
 - address
 - customer
 - customerorderhistory
 - customertypes

 - Columns
 - pk_CustomerType_id
 - CustomerType_name
 - CustomerType_delete_flag

 - Indexes
 - Foreign Keys
 - Triggers
 - department
 - discountcoupon
 - employee
 - employeesalary

Information

No object selected

Object Info Session

Query Completed

customer_view_2 customer_view_3 customer_view_2 customer_view_4 customer_view_3 shipping_employee_view shipping_employee_view_2 Procedures* SQL File 44* SQL File 44*

```
1  select * from department;
```

pk_Department_id	Department_name	ProductDepartment_delete_flag
1	Produce	F
2	Dairy	F
3	Meat/SeaFood	F
NULL	NULL	NULL

Result Set Filter: Edit Export/Import Wrap Cell Content:

department 16 ×

Action Output

Time	Action	Message	Duration / Fetch
354 14:18:22	select * from grocerystore LIMIT 0, 1000	3 row(s) returned	0.031 sec / 0.000 sec
355 14:18:49	CALL insert_product_data ('mushrooms', 'Organic Mushrooms', 1, 3.21, 'produce', 10, 'Whole Foods')	1 row(s) affected	0.094 sec
356 14:18:56	select * from grocerystore LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec
357 14:19:10	select * from product LIMIT 0, 1000	7 row(s) returned	0.000 sec / 0.000 sec
358 14:21:13	select * from productavailability LIMIT 0, 1000	13 row(s) returned	0.016 sec / 0.000 sec
359 14:22:10	select * from department LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec

Output

Apply Cancel

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator

MANAGEMENT

- Server Status
- Client Connections
- Users and Privileges
- Status and System Variables
- Data Export
- Data Import/Restore

INSTANCE

- Startup / Shutdown
- Server Logs
- Options File

SCHEMAS

- Filter objects
- instacart_db

 - Tables
 - address
 - customer
 - customerorderhistory
 - customertypes

 - Columns
 - pk_CustomerType_id
 - CustomerType_name
 - CustomerType_delete_flag

 - Indexes
 - Foreign Keys
 - Triggers
 - department
 - discountcoupon
 - employee
 - employeesalary

Information

No object selected

Object Info Session

Query Completed

customer_view_2 customer_view_3 customer_view_2 customer_view_4 customer_view_3 shipping_employee_view shipping_employee_view_2 Procedures* SQL File 44* SQL File 44*

```
1  select * from grocerystore;
```

pk_GroceryStore_id	GroceryStore_name	GroceryStore_delete_flag
1	Whole Foods	F
2	Costa Wholesale	F
3	Shaws	F
NULL	NULL	NULL

Result Set Filter: Edit Export/Import Wrap Cell Content:

grocerystore 17 ×

Action Output

Time	Action	Message	Duration / Fetch
355 14:18:49	CALL insert_product_data ('mushrooms', 'Organic Mushrooms', 1, 3.21, 'produce', 10, 'Whole Foods')	1 row(s) affected	0.094 sec
356 14:18:56	select * from grocerystore LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec
357 14:19:10	select * from product LIMIT 0, 1000	7 row(s) returned	0.000 sec / 0.000 sec
358 14:21:13	select * from productavailability LIMIT 0, 1000	13 row(s) returned	0.016 sec / 0.000 sec
359 14:22:10	select * from department LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec
360 14:23:42	select * from grocerystore LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec

Output

Apply Cancel

B) Stored Procedure created to enter the person data based on the type of person:

DELIMITER \$\$\$

```
CREATE PROCEDURE Insert_person_data_2 (first_name varchar(45), middle_name varchar(45),
last_name varchar(45), age tinyint(4), gender varchar(10),
email_id varchar(70), phone_no varchar(10), street_addr varchar(70), apt_no varchar(30), city
varchar(45), state varchar(45), zip varchar(9),
username varchar(20), pass varchar(20), person_type char(1), customer_type varchar(10), emp_title
varchar(30), date_of_joining date, salary float )
```

BEGIN

```
INSERT INTO person values(null, first_name, middle_name, last_name, age, gender,'F');
```

```
SET @person_id = (SELECT MAX(pk_Person_id) FROM PERSON);
```

```
INSERT INTO personemail values (null,@person_id, email_id, 'F');
```

```
INSERT INTO personphone values (null, @person_id, Phone_number, 'F');
```

```
INSERT INTO address values (null, street_addr, apt_no, city, state, zip, 'F');
```

```
SET @address_id =(SELECT MAX(pk_Address_id) FROM address);
```

```
INSERT INTO `person/address` values(null,@address_id,@person_id, 'F');
```

```
INSERT INTO personaccount values (null, @person_id, username, pass , 'F');
```

```
IF (person_type = 'c')
```

THEN

```
SET @c_type_id = (SELECT pk_CustomerType_id FROM customertype WHERE
CustomerType_name=customer_type);
```

```
INSERT INTO customer values (null, now(), @person_id, @c_type_id, 'F');
```

ELSE IF(person_type = 'e')

THEN

```
INSERT INTO employee values(null,@person_id,emp_title,date_of_joining, null, 'F', null);
```

```
SET @emp_id = (SELECT MAX(pk_Employee_id) FROM employee);
```

```
INSERT INTO employeesalary values (null, @emp_id, salary, 'F');
```

```
END IF;
```

```
END IF;
```

```
END; $$
```

DBMS Project - Fall 2014

```

MySQL Workbench
File Edit View Query Database Server Tools Scripting Help
customer_view_2 customer_view_3 customer_view_2 customer_view_4 customer_view_3 shipping_employee_view shipping_employee_view_2 Procedures* SQL File 43 SQL File 44*
MANAGEMENT
    Server Status
    Client Connections
    Users and Privileges
    Status and System Variables
    Data Export
    Data Import/Restore
INSTANCE
    Startup / Shutdown
    Server Logs
    Options File
SCHEMAS
    instacart_db
        Tables
        address
        customer
        customerorderhistory
        customertype
            Columns
                pk_CustomerType_id
                CustomerType_name
                CustomerType_delete_flag
            Indices
            Foreign Keys
            Triggers
        department
        discountcoupon
        employee
        employeesalary
Information
No object selected
Object Info Session
Query Completed
DELIMITER $$$
CREATE PROCEDURE Insert_person_data_2(first_name varchar(45), middle_name varchar(45), last_name varchar(45), age tinyint, gender varchar(10),
@email_id varchar(20), phone_no varchar(10), street_addr varchar(20), apt_no varchar(10), city varchar(20), state varchar(10), zip varchar(10),
username varchar(20), pass varchar(20), person_type char(1), customer_type varchar(10), emp_title varchar(20), date_of_joining date, salary float )
BEGIN
    INSERT INTO person values(null, first_name, middle_name, last_name, age, gender, 'F');
    SET @person_id = (SELECT MAX(pk_Person_id) FROM PERSON);
    INSERT INTO personemail values (null,@person_id,@email_id,'F');
    INSERT INTO personphone values (null,@person_id,Phone_number, 'F');
    INSERT INTO personaddress values (null,@person_id,street_addr,apt_no,city,state,zip, 'F');
    SET @address_id = (SELECT MAX(pk_Address_id) FROM ADDRESS);
    INSERT INTO `person/address` values(null,@address_id,@person_id, 'F');
    INSERT INTO personaccount values (null,@person_id,username,pass, 'F');
    IF (person_type = 'c')
    THEN
        SET @c_type_id = (SELECT pk_CustomerType_id FROM customertype WHERE CustomerType_name=customer_type);
        INSERT INTO customer values (null,@person_id,@c_type_id, 'F');
    ELSE IF(person_type = 'e')
    THEN
        INSERT INTO employee values(null,@person_id,emp_title,date_of_joining, null, 'F', null);
        SET @emp_id = (SELECT MAX(pk_Employee_id) FROM employee);
        INSERT INTO employeesalary values (null,@emp_id,salary, 'F');
    END IF;
    END IF;
END $$
    
```

The screenshot shows the MySQL Workbench interface with the 'Procedures*' tab selected. The code editor contains the creation of a stored procedure 'Insert_person_data_2'. The procedure takes various parameters including names, age, gender, email, phone number, address details, password, user type, job title, joining date, and salary. It uses the 'DELIMITER \$\$' command to define the procedure body. Inside the body, it inserts records into the 'person', 'personemail', 'personphone', 'personaddress', 'personaccount', and 'customer' or 'employee' tables based on the 'person_type' parameter. The 'person_id' is determined by a MAX query on the 'PERSON' table. The 'address_id' is determined by a MAX query on the 'ADDRESS' table. The 'c_type_id' is determined by a MAX query on the 'customertype' table. The 'emp_id' is determined by a MAX query on the 'employee' table. The 'salary' value is set to null for customers.

Calling the procedure:

1) When person is of type customer:

```

MySQL Workbench
File Edit View Query Database Server Tools Scripting Help
customer_view_2 customer_view_3 customer_view_2 customer_view_4 customer_view_3 shipping_employee_view shipping_employee_view_2 Procedures* SQL File 43 SQL File 44*
MANAGEMENT
    Server Status
    Client Connections
    Users and Privileges
    Status and System Variables
    Data Export
    Data Import/Restore
INSTANCE
    Startup / Shutdown
    Server Logs
    Options File
SCHEMAS
    instacart_db
        Tables
        address
        customer
        customerorderhistory
        customertype
            Columns
                pk_CustomerType_id
                CustomerType_name
                CustomerType_delete_flag
            Indices
            Foreign Keys
            Triggers
        department
        discountcoupon
        employee
        employeesalary
Information
No object selected
Object Info Session
Query Completed
CALL Insert_person_data_2('harshit','k','hubert',26,'Male','harsshit@gmail.com','3265789514','132 Louis Lane', 'NY', '10001, 'harshit','56Bar09','c','express',null,null,null)
    
```

The screenshot shows the MySQL Workbench interface with the 'SQL File 44*' tab selected. The code editor contains a single call to the 'Insert_person_data_2' procedure. The parameters passed are: first_name='harshit', middle_name='k', last_name='hubert', age=26, gender='Male', email='harsshit@gmail.com', address_id='3265789514', street_addr='132 Louis Lane', city='NY', zip='10001', username='harshit', pass='56Bar09', person_type='c', customer_type='express', emp_title=null, date_of_joining=null, salary=null. The output pane shows the execution log with the message '1 row(s) affected'.

DBMS Project - Fall 2014

Updated Tables:

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator

MANAGEMENT

- Server Status
- Client Connections
- Users and Privileges
- Status and System Variables
- Data Export
- Data Import/Restore

INSTANCE

- Startup / Shutdown
- Server Logs
- Options File

SCHEMAS

instacart_db

- Tables
- customer
- customerorderhistory
- customertypes
- customer
- customer_type
- customer_type_id
- CustomerType_name
- CustomerType_delete_flag
- Indexes
- Foreign Keys
- Triggers
- department
- discountcoupon
- employee
- employeesalary

Information

No object selected

Object Info Session

Query Completed

```
ew customer_view_2 customer_view_3 customer_view_2 customer_view_4 customer_view_3 shipping_employee_view shipping_employee_view_2 Procedures* SQL File 43 SQL File 44*
```

Result Set Filter: Edit Export/Import Wrap Cell Content

1 select * from persons

	pk_Person_id	first_name	middle_name	last_name	age	gender	Person_delete_flag
1	1	Valbhav	R	Mistry	24	Male	F
2	2	Richa	P	Sharma	19	Female	F
3	3	Ankit	E	Gala	42	Male	F
4	4	Jimmy	H	Shah	36	Male	F
5	5	Sarah	J	Feller	50	Female	F
6	6	hanshit	k	patel	26	Male	F
7	7	hanshit	k	hubert	26	Male	F
8	8	hanshit	K	hubert	26	Male	F
*							

person 4

Action Output

Time	Action	Message	Duration / Fetch
339	13:59:44 CALL Insert_person_data_2('hanshit', 'K', 'patel', 26, 'Male', 'hanshit@gmail.com', 3265789514, '132 Louis Lane')	Error Code: 1062. Duplicate entry 'patel' for key 'last_name_idx'	0.047 sec
340	14:00:34 CALL Insert_person_data_2('hanshit', 'K', 'hubert', 26, 'Male', 'hanshit@gmail.com', 3265789514, '132 Louis Lane')	1 row(s) affected	2.277 sec
341	14:01:44 SELECT * from person LIMIT 0, 1000	7 row(s) returned	0.000 sec / 0.000 sec
342	14:02:10 SELECT * from address LIMIT 0, 1000	10 row(s) returned	0.016 sec / 0.000 sec
343	14:02:42 SELECT * from customer LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec
344	14:05:42 select *from person LIMIT 0, 1000	7 row(s) returned	0.000 sec / 0.000 sec

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator

MANAGEMENT

- Server Status
- Client Connections
- Users and Privileges
- Status and System Variables
- Data Export
- Data Import/Restore

INSTANCE

- Startup / Shutdown
- Server Logs
- Options File

SCHEMAS

instacart_db

- Tables
- customer
- customerorderhistory
- customertypes
- customer
- customer_type
- customer_type_id
- CustomerType_name
- CustomerType_delete_flag
- Indexes
- Foreign Keys
- Triggers
- department
- discountcoupon
- employee
- employeesalary

Information

No object selected

Object Info Session

Query Completed

```
ew customer_view_2 customer_view_3 customer_view_2 customer_view_4 customer_view_3 shipping_employee_view shipping_employee_view_2 Procedures* SQL File 43 SQL File 44*
```

Result Set Filter: Edit Export/Import Wrap Cell Content

1 select * from address

	pk_Address_id	Street_name	Apt_number	Cty	State	Zip	Address_delete_flag
5	181 hemenway st	8	boston	MA	2115	F	
6	132 Pikenng st	2	San Jose	CA	95103	F	
7	162 hemenway st	B2	Boston	MA	2115	F	
8	795 E DRAGRAM		TUCSON	AZ	85105	F	
9	300 BOYLSTON AVE		SEATTLE	WA	98102	F	
10	2430 Naudkott Place		Washington	DC	20521	F	
11	6606 Tusing Road		Reynoldsburg	OH	43068	F	
12	175 hemenway st		Boston	MA	2115	F	
13	132 Louis Lane		New York	NY	10001	F	
*							

address 5

Action Output

Time	Action	Message	Duration / Fetch
340	14:00:34 CALL Insert_person_data_2('hanshit', 'K', 'hubert', 26, 'Male', 'hanshit@gmail.com', 3265789514, '132 Louis Lane')	1 row(s) affected	2.277 sec
341	14:01:44 SELECT * from person LIMIT 0, 1000	7 row(s) returned	0.000 sec / 0.000 sec
342	14:02:10 SELECT * from address LIMIT 0, 1000	10 row(s) returned	0.016 sec / 0.000 sec
343	14:02:42 SELECT * from customer LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec
344	14:05:42 select *from person LIMIT 0, 1000	7 row(s) returned	0.000 sec / 0.000 sec
345	14:06:45 select *from address LIMIT 0, 1000	10 row(s) returned	0.000 sec / 0.000 sec

DBMS Project - Fall 2014

MySQL Workbench

MySQL Model "Instacart_project..." EER Diagram Local instance MySQL56

File Edit View Query Database Server Tools Scripting Help

Navigator

MANAGEMENT

- Server Status
- Client Connections
- Users and Privileges
- Status and System Variables
- Data Export
- Data Import/Restore

INSTANCE

- Startup / Shutdown
- Server Logs
- Options File

SCHEMAS

- instacart_db
 - Tables
 - customer
 - customerorderhistory
 - customertypes
 - customer
 - address
 - department
 - discountcoupon
 - employee
 - employeesalary

Information

No object selected

Object Info Session

Query Completed

customer_view_2 customer_view_3 customer_view_2 customer_view_4 customer_view_3 shipping_employee_view shipping_employee_view_2 Procedures* SQL File 43* SQL File 44*

Result Set Filter

```
1 select * from customer;
```

	pk_Customer_id	ActiveFrom	Rk_person_id	Rk_Customertype_id	Customer_delete_flag
1	2014-01-10 01:02:00	1	1	F	
2	2012-08-08 03:02:00	2	2	F	
3	2011-03-20 03:02:00	3	1	F	
4	2014-12-02 14:00:35	8	2	F	
*	NULL	NULL	NULL	NULL	

customer 6 x

Output

Action Output

Time	Action	Message	Duration / Fetch
341 14:01:44	SELECT * from person LIMIT 0, 1000	7 row(s) returned	0.000 sec / 0.000 sec
342 14:02:10	SELECT * from Address LIMIT 0, 1000	10 row(s) returned	0.016 sec / 0.000 sec
343 14:02:42	SELECT * from customer LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec
344 14:05:42	select *from person LIMIT 0, 1000	7 row(s) returned	0.000 sec / 0.000 sec
345 14:06:45	select *from address LIMIT 0, 1000	10 row(s) returned	0.000 sec / 0.000 sec
346 14:07:30	select *from customer LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec

MySQL Workbench

MySQL Model "Instacart_project..." EER Diagram Local instance MySQL56

File Edit View Query Database Server Tools Scripting Help

Navigator

MANAGEMENT

- Server Status
- Client Connections
- Users and Privileges
- Status and System Variables
- Data Export
- Data Import/Restore

INSTANCE

- Startup / Shutdown
- Server Logs
- Options File

SCHEMAS

- instacart_db
 - Tables
 - customer
 - customerorderhistory
 - customertypes
 - customer
 - address
 - department
 - discountcoupon
 - employee
 - employeesalary

Information

No object selected

Object Info Session

Query Completed

customer_view_2 customer_view_3 customer_view_2 customer_view_4 customer_view_3 shipping_employee_view shipping_employee_view_2 Procedures* SQL File 43* SQL File 44*

Result Set Filter

```
1 select * from personaccounts;
```

	pk_PersonAccount_id	Person_Person_id	username	password	PersonAccount_delete_flag
1	1	vahbav	abc123	F	
2	2	richa	12b3	F	
3	3	arkits	sub1256	F	
4	4	jim	56tty	F	
5	5	sarah	gf456	F	
6	6	hashit	59har09	F	

personaccount 7 x

Output

Action Output

Time	Action	Message	Duration / Fetch
342 14:02:10	SELECT * from Address LIMIT 0, 1000	10 row(s) returned	0.016 sec / 0.000 sec
343 14:02:42	SELECT * from customer LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec
344 14:05:42	select *from person LIMIT 0, 1000	7 row(s) returned	0.000 sec / 0.000 sec
345 14:06:45	select *from address LIMIT 0, 1000	10 row(s) returned	0.000 sec / 0.000 sec
346 14:07:30	select *from customer LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec
347 14:10:57	select *from personaccount LIMIT 0, 1000	7 row(s) returned	0.000 sec / 0.000 sec

DBMS Project - Fall 2014

2) Calling the procedure when the person is of type employee

The screenshot shows the MySQL Workbench interface. In the center pane, a SQL query is being run:

```
CALL Insert_person_data_2 ('Shally', null, 'Arona', 41, 'Female', 'shally@hotmail.com', 9856321451, '555 Dorchester Ave', null, 'Dorchester', 'NJ', 05117, 'shally', 'shally654', 'e', null, 'Shipping Employee', '2013-02-06', 50000);
```

The Output pane displays the results of the query, showing the execution log:

Time	Action	Message	Duration / Fetch
344 14:06:42	select * from person LIMIT 0, 1000	7 row(s) returned	0.000 sec / 0.000 sec
345 14:06:45	select * from address LIMIT 0, 1000	10 row(s) returned	0.000 sec / 0.000 sec
346 14:07:30	select * from customer LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec
347 14:10:57	select * from personaccount LIMIT 0, 1000	7 row(s) returned	0.000 sec / 0.000 sec
348 14:12:46	CALL Insert_person_data_2('Shally', null, 'Arona', 41, 'Female', 'shally@hotmail.com', 9856321451, '555 D...', Error Code: 1054. Unknown column 'Dorchester' in field list'	1 row(s) affected	0.000 sec
349 14:13:05	CALL Insert_person_data_2('Shally', null, 'Arona', 41, 'Female', 'shally@hotmail.com', 9856321451, '555 D...', 1 row(s) affected)	0.297 sec	

The updated tables are as below:

The screenshot shows the MySQL Workbench interface. A query is run against the personaccount table:

```
select * from personaccount;
```

The Result Set Filter pane shows the data:

pk_PersonAccount_id	Person_Person_id	username	password	PersonAccount_delete_flag
1	1	vabnav	abc123	F
2	2	richa	12h3	F
3	3	ankits	sub1256	F
4	4	jim	56ity	F
5	5	sarah	ge456	F
6	6	harshit	59har09	F
7	8	harshit	59har09	F
8	9	shally	shally654	F
*	HULL	HULL	HULL	HULL

The Output pane displays the results of the query, showing the execution log:

Time	Action	Message	Duration / Fetch
345 14:06:45	select * from address LIMIT 0, 1000	10 row(s) returned	0.000 sec / 0.000 sec
346 14:07:30	select * from customer LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec
347 14:10:57	select * from personaccount LIMIT 0, 1000	7 row(s) returned	0.000 sec / 0.000 sec
348 14:12:46	CALL Insert_person_data_2('Shally', null, 'Arona', 41, 'Female', 'shally@hotmail.com', 9856321451, '555 D...', Error Code: 1054. Unknown column 'Dorchester' in field list'	1 row(s) affected	0.000 sec
349 14:13:05	CALL Insert_person_data_2('Shally', null, 'Arona', 41, 'Female', 'shally@hotmail.com', 9856321451, '555 D...', 1 row(s) affected)	0.297 sec	
350 14:13:45	select * from personaccount LIMIT 0, 1000	8 row(s) returned	0.000 sec / 0.000 sec

DBMS Project - Fall 2014

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator

MANAGEMENT

- Server Status
- Client Connections
- Users and Privileges
- Status and System Variables
- Data Export
- Data Import/Restore

INSTANCE

- Startup / Shutdown
- Server Logs
- Options File

SCHEMAS

- instacart_db
 - Tables
 - address
 - customer
 - customerorderhistory
 - customertypes
 - Columns
 - pk_CustomerType_id
 - CustomerType_name
 - CustomerType_delete_flag
 - Indexes
 - Foreign Keys
 - Triggers
 - department
 - discountcoupon
 - employee
 - employeesalary

Information

No object selected

Object Info Session

Query Completed

Result Set Filter: Wrap Cell Content:

```
1 * select * from persons;
```

	pk_Person_id	first_name	middle_name	last_name	age	gender	Person_delete_flag
1	Valbhav	R		Matty	24	Male	F
2	Richa	P		Sharma	19	Female	F
3	Ankit	E		Gala	42	Male	F
4	Jimmy			Shah	36	Male	F
5	Sarah	J		Feller	50	Female	F
6	hansht	k		patel	26	Male	F
8	hansht	k		hubert	26	Male	F
9	Shaly			Arona	41	Female	F
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

person 9 < x

Action Output

Time	Action	Message	Duration / Fetch
346 14:07:30	select * from customer LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec
347 14:10:57	select * from personaccount LIMIT 0, 1000	7 row(s) returned	0.000 sec / 0.000 sec
348 14:12:46	CALL Insert_person_data_2('Shaly', null, 'Arona', 41, 'Female', 'shaly@hotmail.com', 3856321451, 555 D...	Error Code: 1054. Unknown column 'Dorchester' in field list'	0.000 sec
349 14:13:05	CALL Insert_person_data_2('Shaly', null, 'Arona', 41, 'Female', 'shaly@hotmail.com', 3856321451, 555 D...	1 row(s) affected	0.297 sec
350 14:13:45	select * from personaccount LIMIT 0, 1000	8 row(s) returned	0.000 sec / 0.000 sec
351 14:14:39	select * from person LIMIT 0, 1000	8 row(s) returned	0.000 sec / 0.000 sec
351 14:14:39	select * from employee LIMIT 0, 1000	8 row(s) returned	0.000 sec / 0.000 sec

Output

Query Completed

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator

MANAGEMENT

- Server Status
- Client Connections
- Users and Privileges
- Status and System Variables
- Data Export
- Data Import/Restore

INSTANCE

- Startup / Shutdown
- Server Logs
- Options File

SCHEMAS

- instacart_db
 - Tables
 - address
 - customer
 - customerorderhistory
 - customertypes
 - Columns
 - pk_CustomerType_id
 - CustomerType_name
 - CustomerType_delete_flag
 - Indexes
 - Foreign Keys
 - Triggers
 - department
 - discountcoupon
 - employee
 - employeesalary

Information

No object selected

Object Info Session

Query Completed

Result Set Filter: Wrap Cell Content:

```
1 * select * from employees;
```

	pk_Employee_id	fk_Person_id	Title	date_of_joining	date_of_leaving	Employee_delete_flag	Employee_manager_id
1	4	Manager		2005-01-18	NULL	F	1
2	5	Shipping Employee	2012-05-08	NULL	NULL	F	1
3	9	Shipping Employee	2013-02-06	NULL	NULL	F	NULL
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

employee 10 < x

Action Output

Time	Action	Message	Duration / Fetch
347 14:10:57	select * from personaccount LIMIT 0, 1000	7 row(s) returned	0.000 sec / 0.000 sec
348 14:12:46	CALL Insert_person_data_2('Shaly', null, 'Arona', 41, 'Female', 'shaly@hotmail.com', 3856321451, 555 D...	Error Code: 1054. Unknown column 'Dorchester' in field list'	0.000 sec
349 14:13:05	CALL Insert_person_data_2('Shaly', null, 'Arona', 41, 'Female', 'shaly@hotmail.com', 3856321451, 555 D...	1 row(s) affected	0.297 sec
350 14:13:45	select * from personaccount LIMIT 0, 1000	8 row(s) returned	0.000 sec / 0.000 sec
351 14:14:30	select * from person LIMIT 0, 1000	8 row(s) returned	0.000 sec / 0.000 sec
352 14:15:12	select * from employee LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec

Output

Query Completed

10. Triggers

1) To create a backup of product:

a) ON UPDATE:

```
DELIMITER $$$
CREATE TRIGGER product_backup_t
BEFORE UPDATE
ON product
FOR EACH ROW
BEGIN
SET @prod_id = pk_Product_id;

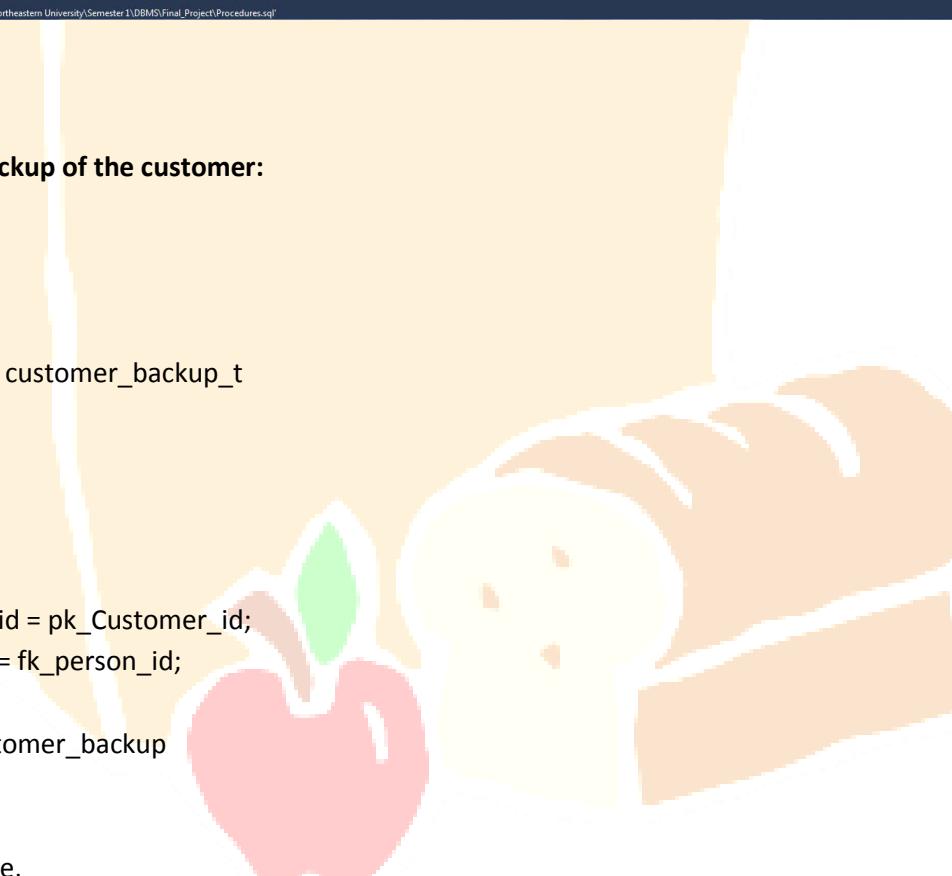
INSERT INTO product_backup
SELECT @prod_id,
product.Product_name,
product.Product_description,
product.Product_weight,
product.Product_price,
department.Department_name
FROM product
INNER JOIN department
ON product.fk_Department_id = department.pk_Department_id;
END; $$
```

b) ON DELETE

```
DELIMITER $$$
CREATE TRIGGER product_backup_bdelete_t
BEFORE DELETE
ON product
FOR EACH ROW
BEGIN
SET @prod_id = pk_Product_id;

INSERT INTO product_backup
SELECT @prod_id,
product.Product_name,
product.Product_description,
product.Product_weight,
product.Product_price,
department.Department_name
FROM product
INNER JOIN department
```

```
ON product.fk_Department_id = department.pk_Department_id;
END; $$$
```



The screenshot shows the MySQL Workbench interface. In the center, there is a large code editor window displaying SQL triggers for the 'product' table. The triggers are named 'product_backup_t' and 'product_backup_bdelete_t'. The code includes logic to insert data into a backup table ('customer_backup') before an update or delete operation on the 'product' table. On the left side, the Navigator pane shows the database schema, including tables like 'customer', 'person', and 'department'. The bottom left corner of the interface has a status bar with the path 'C:\Users\Vaibhav\Desktop\Northeastern University\Semester 1\DBMS\Final Project\Procedures.sql'.

```

DELIMITER $$$
CREATE TRIGGER product_backup_t
BEFORE UPDATE
ON product
FOR EACH ROW
BEGIN
    SET @prod_id = pk_Product_id;
    INSERT INTO product_backup
    SELECT @prod_id,
    product.Product_name,
    product.Product_description,
    product.Product_weight,
    product.Product_price,
    department.Department_name
    FROM product
    INNER JOIN department
    ON product.fk_Department_id = department.pk_Department_id;
END$$
DELIMITER $$$
CREATE TRIGGER product_backup_bdelete_t
BEFORE DELETE
ON product
FOR EACH ROW
BEGIN
    SET @prod_id = pk_Product_id;
    INSERT INTO product_backup
    SELECT @prod_id,
    product.Product_name,
    product.Product_description,
    product.Product_weight,
    product.Product_price,
    department.Department_name
    FROM product
    INNER JOIN department
    ON product.fk_Department_id = department.pk_Department_id;
END$$

```

2) To create a backup of the customer:

A) ON UPDATE

DELIMITER !!!

```
CREATE TRIGGER customer_backup_t
BEFORE UPDATE
ON customer
FOR EACH ROW
BEGIN
```

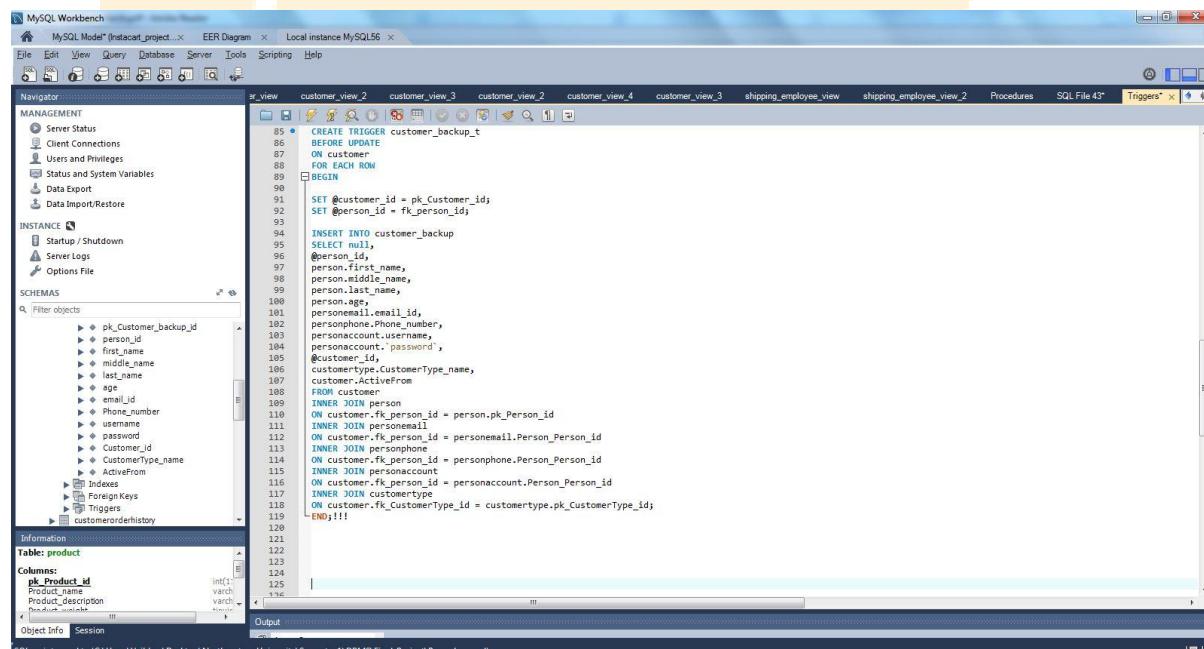
```
SET @customer_id = pk_Customer_id;
SET @person_id = fk_person_id;
```

```
INSERT INTO customer_backup
SELECT null,
@person_id,
person.first_name,
person.middle_name,
person.last_name,
person.age,
personemail.email_id,
personphone.Phone_number,
```

```

personaccount.username,
personaccount.`password`,
@customer_id,
customertype.CustomerType_name,
customer.ActiveFrom
FROM customer
INNER JOIN person
ON customer.fk_person_id = person.pk_Person_id
INNER JOIN personemail
ON customer.fk_person_id = personemail.Person_Person_id
INNER JOIN personphone
ON customer.fk_person_id = personphone.Person_Person_id
INNER JOIN personaccount
ON customer.fk_person_id = personaccount.Person_Person_id
INNER JOIN customertype
ON customer.fk_CustomerType_id = customertype.pk_CustomerType_id;
END;!!!

```



b) ON DELETE

DELIMITER !!!

```

CREATE TRIGGER customer_backup_bdelete_t
BEFORE DELETE
ON customer
FOR EACH ROW
BEGIN

```

```

SET @customer_id = pk_Customer_id;
SET @person_id = fk_person_id;

INSERT INTO customer_backup
SELECT null,
@person_id,
person.first_name,
person.middle_name,
person.last_name,
person.age,
personemail.email_id,
personphone.Phone_number,
personaccount.username,
personaccount.`password`,
@customer_id,
customertype.CustomerType_name,
customer.ActiveFrom
FROM customer
INNER JOIN person
ON customer.fk_person_id = person.pk_Person_id
INNER JOIN personemail
ON customer.fk_person_id = personemail.Person_Person_id
INNER JOIN personphone
ON customer.fk_person_id = personphone.Person_Person_id
INNER JOIN personaccount
ON customer.fk_person_id = personaccount.Person_Person_id
INNER JOIN customertype
ON customer.fk_CustomerType_id = customertype.pk_CustomerType_id;
END;!!!

```



The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** Shows the database structure with tables like `customer`, `person`, `personemail`, `personphone`, `personaccount`, and `customertype`.
- SQL Editor:** Displays the complete SQL script for creating a trigger named `customer_backup_bdelete_t`.
- Output:** Shows the results of the executed SQL statements.
- Information:** Provides details about the `product` table, including its columns and data types.
- Object Info:** Shows the properties of the current selected object.
- Session:** Displays session information.

```

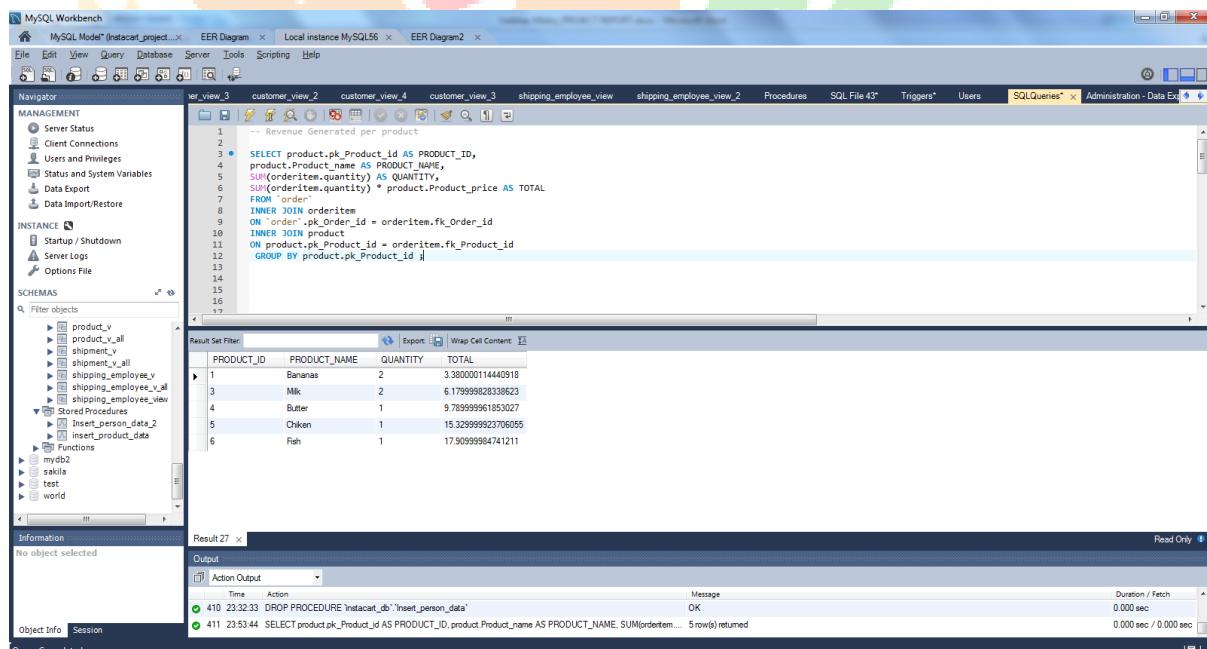
DELIMITER //
CREATE TRIGGER customer_backup_bdelete_t
BEFORE DELETE
ON customer
FOR EACH ROW
BEGIN
    SET @customer_id = pk_Customer_id;
    SET @person_id = fk_person_id;
    INSERT INTO customer_backup
    SELECT null,
    @person_id,
    person.first_name,
    person.middle_name,
    person.last_name,
    person.age,
    personemail.email_id,
    personphone.Phone_number,
    personaccount.username,
    personaccount.`password`,
    @customer_id,
    customertype.CustomerType_name,
    customer.ActiveFrom
    FROM customer
    INNER JOIN person
    ON customer.fk_person_id = person.pk_Person_id
    INNER JOIN personemail
    ON customer.fk_person_id = personemail.Person_Person_id
    INNER JOIN personphone
    ON customer.fk_person_id = personphone.Person_Person_id
    INNER JOIN personaccount
    ON customer.fk_person_id = personaccount.Person_Person_id
    INNER JOIN customertype
    ON customer.fk_CustomerType_id = customertype.pk_CustomerType_id;
    ON customer.fk_CustomerType_id = customertype.pk_CustomerType_id;
END;!!!

```

11. High Level Questions

A) To find out the revenue generated per product:

```
SELECT product.pk_Product_id AS PRODUCT_ID,
product.Product_name AS PRODUCT_NAME,
SUM(orderitem.quantity) AS QUANTITY,
SUM(orderitem.quantity) * product.Product_price AS TOTAL
FROM `order`
INNER JOIN orderitem
ON `order`.pk_Order_id = orderitem.fk_Order_id
INNER JOIN product
ON product.pk_Product_id = orderitem.fk_Product_id
GROUP BY product.pk_Product_id ;
```



The screenshot shows the MySQL Workbench interface. The SQL editor tab contains the query provided above. The results are displayed in a table titled 'Result Set Filter'.

PRODUCT_ID	PRODUCT_NAME	QUANTITY	TOTAL
1	Bananas	2	3.38000114440918
3	Milk	2	6.179999628338623
4	Butter	1	9.789999951615027
5	Chiken	1	15.329999923706055
6	Fish	1	17.90999984741211

The 'Output' pane at the bottom shows the execution log:

- 410 23:32:33 DROP PROCEDURE 'instacart_db`.`Insert_person_data'
- 411 23:53:44 SELECT product.pk_Product_id AS PRODUCT_ID, product.Product_name AS PRODUCT_NAME, SUM(orderitem.quantity) AS QUANTITY, SUM(orderitem.quantity) * product.Product_price AS TOTAL

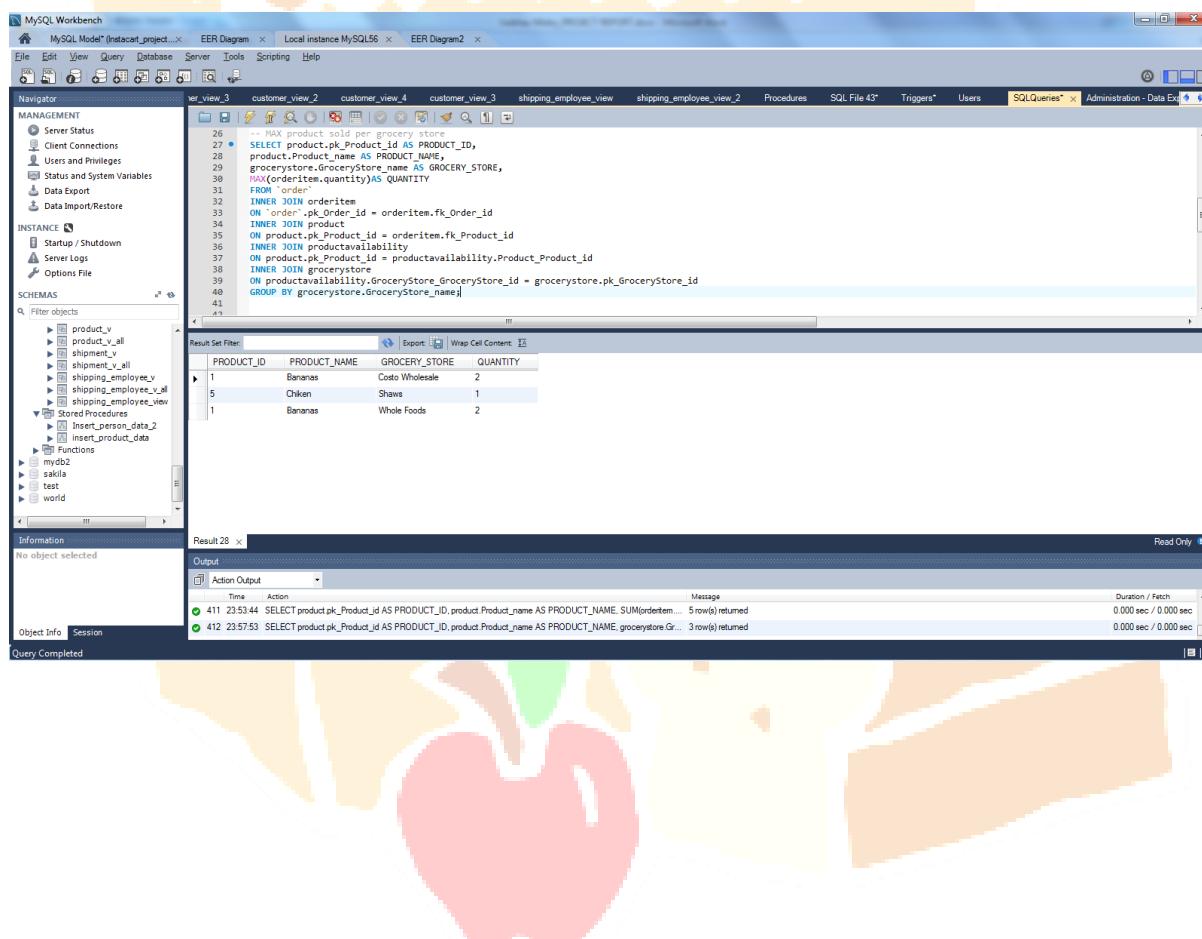
B) To find out the maximum product sold per Grocery Store:

```

SELECT product.pk_Product_id AS PRODUCT_ID,
product.Product_name AS PRODUCT_NAME,
grocerystore.GroceryStore_name AS GROCERY_STORE,
MAX(orderitem.quantity)AS QUANTITY
FROM `order`  

INNER JOIN orderitem
ON `order`.pk_Order_id = orderitem.fk_Order_id
INNER JOIN product
ON product.pk_Product_id = orderitem.fk_Product_id
INNER JOIN productavailability
ON product.pk_Product_id = productavailability.Product_Product_id
INNER JOIN grocerystore
ON productavailability.GroceryStore_GroceryStore_id = grocerystore.pk_GroceryStore_id
GROUP BY grocerystore.GroceryStore_name;

```



The screenshot shows the MySQL Workbench interface. The SQL editor tab contains the query from above. The results are displayed in a table titled 'Result Set Filter' with columns: PRODUCT_ID, PRODUCT_NAME, GROCERY_STORE, and QUANTITY. The data shows three rows: one row for Bananas at Costo Wholesale (quantity 2), one row for Chicken at Shaws (quantity 1), and one row for Bananas at Whole Foods (quantity 2). Below the table, the 'Output' pane shows the execution log with two entries: a SELECT statement for the first row and another for the second row.

PRODUCT_ID	PRODUCT_NAME	GROCERY_STORE	QUANTITY
1	Bananas	Costo Wholesale	2
5	Chicken	Shaws	1
1	Bananas	Whole Foods	2

Output

```

Time Action Message Duration / Fetch
411 23:53:44 SELECT product_pk_Product_id AS PRODUCT_ID, product_Product_name AS PRODUCT_NAME, SUM(orderitem_
5 rows(s) returned 0.000 sec / 0.000 sec
412 23:57:53 SELECT product_pk_Product_id AS PRODUCT_ID, product_Product_name AS PRODUCT_NAME, grocerystore_GroceryS...
3 row(s) returned 0.000 sec / 0.000 sec

```

c) To find out total amount spend by each customer:

```

SELECT customer.pk_Customer_id AS CUSTOMER_ID,
person.first_name AS FIRST_NAME,
person.middle_name AS MIDDLE_NAME,
person.last_name AS LAST_NAME,
SUM(`order`.order_price) AS AMOUNT
FROM customer
INNER JOIN person
ON customer.fk_person_id = person.pk_Person_id
INNER JOIN customerorderhistory
ON customer.pk_Customer_id = customerorderhistory.fk_Customer_id
INNER JOIN `order`
ON `order`.pk_Order_id = customerorderhistory.Order_Order_id
GROUP BY customer.pk_Customer_id;

```

The screenshot shows the MySQL Workbench interface. The SQL editor tab contains the query from above. The results are displayed in a table below:

CUSTOMER_ID	FIRST_NAME	MIDDLE_NAME	LAST_NAME	AMOUNT
1	Vabhan	R	Mistry	21.799999237060547
2	Richa	P	Sharma	35
3	Ankit	E	Gala	9.789999961853027

The output pane shows the execution details:

- Action: SELECT product_pk_Product_id AS PRODUCT_ID, product.Product_name AS PRODUCT_NAME, grocerystore.Gr...
- Action: SELECT customer_pk_Customer_id AS CUSTOMER_ID, person.first_name AS FIRST_NAME, person.middle_name...

12. Conclusion

The report consists all the functional and design specifications explained in detail. Data integrity and security standards were considered while making this application. Business model was understood and based on the requirements gathered application was designed using business rules and constraints.

BIBLIOGRAPHY

Book References

- 1 - Modern Database Management, Jeffrey Hoffer
- 2 – Beginning MySQL, Robert Sheldon
- 3 – Beginning SQL, Paul Wilton
- 4 – Class Slides, Prof. Yusuf Ozbek

Websites

- 1 - www.mysql.com
- 2 - www.google.com
- 3 - www.wikipedia.org

