$(Re)^3$ Reduce | Reuse | Recycle

An overview of the database

Divya Shankar

Sabrina Sok

Table of Contents

Business Description

(Re)^3 is a sustainable/environmental friendly business that will require a database system for storing, manipulating, and retrieving data. (Re)^3 supplies environmental friendly kitchen utensils ranging from glassware to steelware registered customers. The primary customers that (Re)^3 cater to are restaurants, and events. When an order is placed, (Re)^3 charges the customer based on the quantity of items requested and their respective unit cost. For each order placed, (Re)^3 also provides a delivery date (three days after an order has been placed). An event customer will have to provide the time of pick up for the rented items (the company assumes all events last for less than a day) and a restaurant customer will put in a request of pick up date for the rented items. As of now, the company has sixty items belonging to twenty different types of kitchen utensils to be rented out, including glass bowls, water glasses, steel spoons, and wine glasses.

(Re)^3 items are manufactured and cleaned in-house using an assembly line process. To monitor this assembly line and ensure smooth business operations, (Re)^3 has a very diverse and skilled workforce. There are four different types of employees at the company - factory staff, delivery staff, administration staff, and managers. The factory staff working at the manufacturing assembly line are the assembly line design engineers, production workers, inventory analysts, plant operators, and quality inspectors. The factory staff working at the cleaning assembly line are quality inspectors and maintenance engineers. The factory staff at (Re)^3 ensures that the raw materials provided by the suppliers are sufficient to construct the kitchen utensils, and performs quality check on the manufactured items and cleanliness of returned items to classify them as good/bad. The delivery staff facilitate delivery and pick up services associated with each order placed. The administration staff handle accounting documents and feedback from customers associated with an order. The managers oversee all segments of the company and provide supervision to run the business effectively and efficiently.

Database Description

The database for (Re)^3 maintains records of kitchen items being manufactured and the incoming and outgoing activities of the manufactured items. The definitions of our data (attributes for each supertypes and subtypes) are stored centrally in the database to help control for data redundancy. Our database will also provide multiple user views to improve data sharing and prevent duplication of data. We created a total of six views for three different users. The users are administration staff, delivery staff, and factory staff. The delivery staff have views of the delivery details and pickup details; the administration staff have views of order details and feedback details; the factory staff have views of quality check and supply details. Since, (Re)^3 will be catering to a number of customers, the database would serve as an ideal tool for information sharing between (Re)^3 and the customers, and improve/ease the communication.

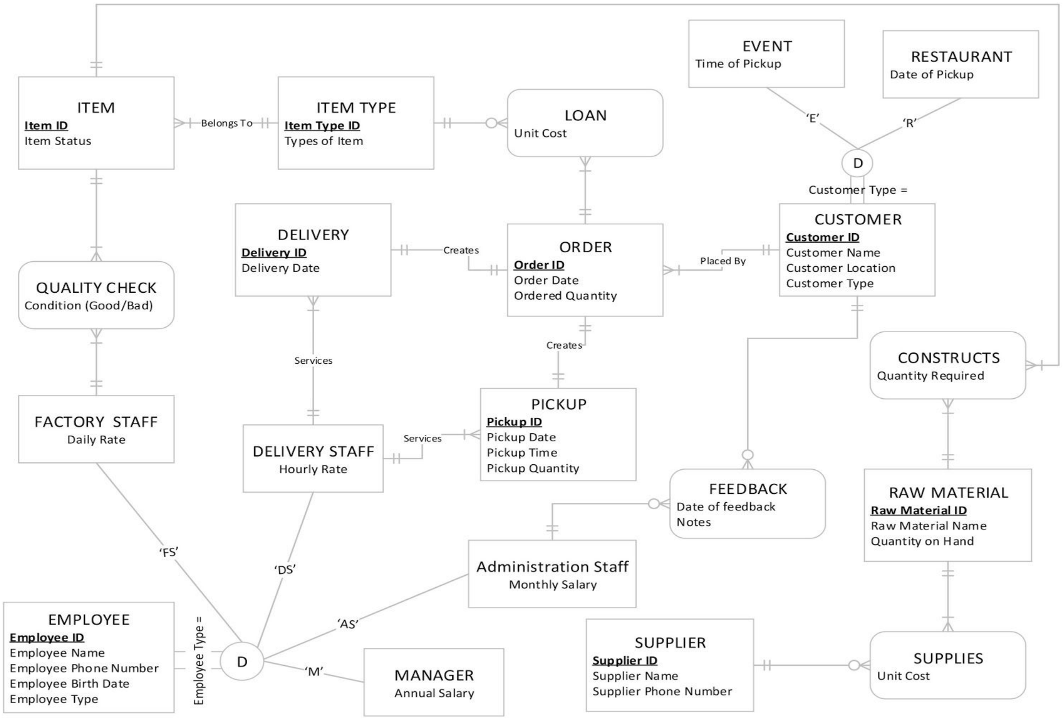
Value of Database

At (Re)^3, there is a large amount of inflow and outflow of transactions. To track each of these transactions efficiently, the company needs a database. Not only will this implementation of database help the company track a large amount of activities, it will ensure consistency across all functions and tables, and also prevent loss of data or misplacement of important data. Since the items at (Re)^3 are rented to customers, manufactured and cleaned in house, it is important to know the status/condition of each item. The company also charges a unit cost for each of the items rented to the customer, and is charged by suppliers for the raw materials used in the manufacturing/cleaning processes. Therefore it is important to have a database at the company to effectively track all the incoming and outgoing data flows linked with manufacturing, cleaning and rental activities. The database would facilitate audit check and keep track of the revenue generated by renting utensils to customers. Lastly, a database system allows the company to create views for specific users to prevent an overwhelming amount of unnecessary data and to help secure data that should not be seen by unauthorized users.

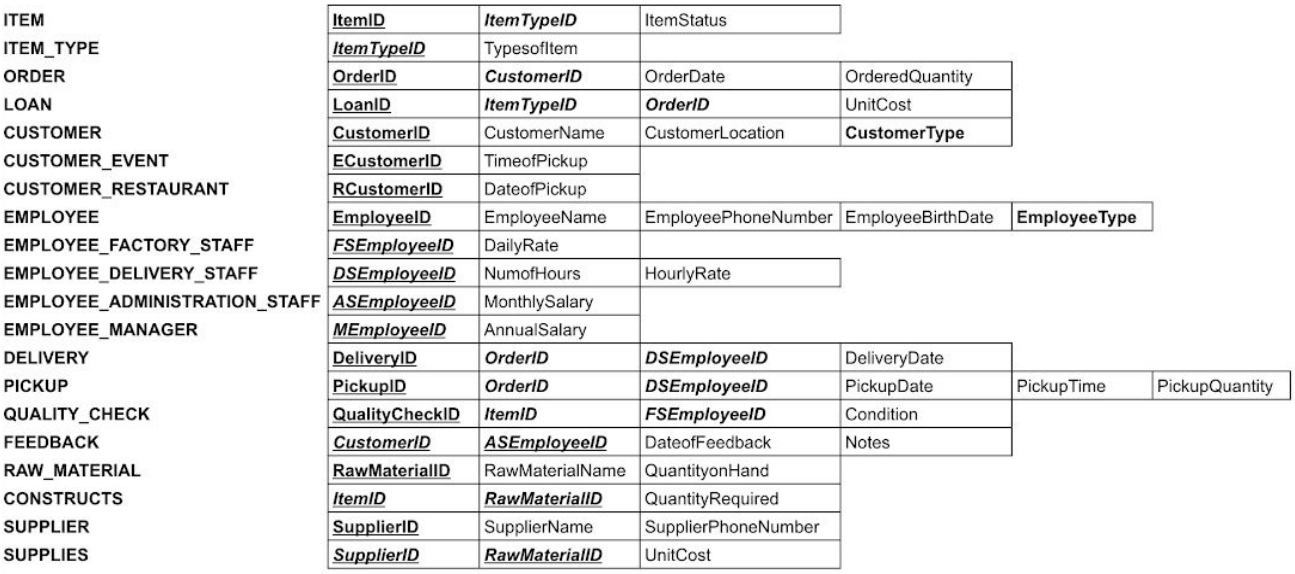
Business Rules

1. Each item at $(Re)^3$ must belong to one and only one item type (eg. item 1 is item type shot glass).
2. A customer will only be recorded in our database after they have placed an order, and the customer must be either a restaurant or an event (disjoint rule).
3. A delivery date has to be created when an order is placed. The delivery date is three days after the a customer creates an order.
4. A pickup date is requested by a customer and inserted into the pickup table. Each of this pickup has to be associated an order from the order table.
5. A factory staff will only be a staff at $(Re)^3$ when they have quality check responsibilities assigned to them.
6. A delivery staff will only be an employee at $(Re)^3$ if they have delivery and/or pickup duties assigned to them.
7. An administration staff will only be an employee at $(Re)^3$ if they have customer feedback handling duties/accounting assigned to them.
8. Employees are divided into four categories: factory staff, delivery staff, administration staff, and manager. Each employee must belong to one and only one staff category.

Conceptual Schema - ER Model

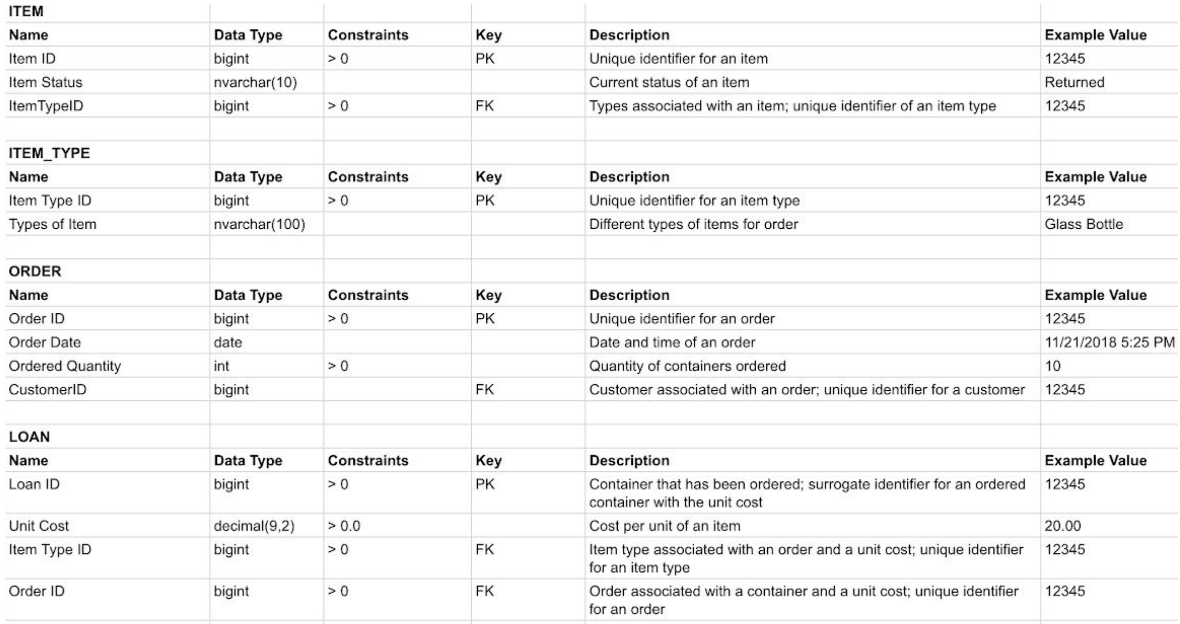


Logical Schema - Relational Table Design



ITEM	**ItemID**	***ItemTypeID***	ItemStatus			
ITEM_TYPE	***ItemTypeID***	TypesofItem				
ORDER	**OrderID**	***CustomerID***	OrderDate	OrderedQuantity		
LOAN	**LoanID**	***ItemTypeID***	***OrderID***	UnitCost		
CUSTOMER	**CustomerID**	CustomerName	CustomerLocation	**CustomerType**		
CUSTOMER_EVENT	**ECustomerID**	TimeofPickup				
CUSTOMER_RESTAURANT	**RCustomerID**	DateofPickup				
EMPLOYEE	**EmployeeID**	EmployeeName	EmployeePhoneNumber	EmployeeBirthDate	**EmployeeType**	
EMPLOYEE_FACTORY_STAFF	***FSEmployeeID***	DailyRate				
EMPLOYEE_DELIVERY_STAFF	***DSEmployeeID***	NumofHours	HourlyRate			
EMPLOYEE_ADMINISTRATION_STAFF	***ASEmployeeID***	MonthlySalary				
EMPLOYEE_MANAGER	***MEmployeeID***	AnnualSalary				
DELIVERY	**DeliveryID**	***OrderID***	***DSEmployeeID***	DeliveryDate		
PICKUP	**PickupID**	***OrderID***	***DSEmployeeID***	PickupDate	PickupTime	PickupQuantity
QUALITY_CHECK	**QualityCheckID**	***ItemID***	***FSEmployeeID***	Condition		
FEEDBACK	***CustomerID***	***ASEmployeeID***	DateofFeedback	Notes		
RAW_MATERIAL	**RawMaterialID**	RawMaterialName	QuantityonHand			
CONSTRUCTS	***ItemID***	***RawMaterialID***	QuantityRequired			
SUPPLIER	**SupplierID**	SupplierName	SupplierPhoneNumber			
SUPPLIES	***SupplierID***	***RawMaterialID***	UnitCost			

Data Dictionary



ITEM

Name	Data Type	Constraints	Key	Description	Example Value
Item ID	bigint	> 0	PK	Unique identifier for an item	12345
Item Status	nvarchar(10)			Current status of an item	Returned
ItemTypeID	bigint	> 0	FK	Types associated with an item; unique identifier of an item type	12345

ITEM_TYPE

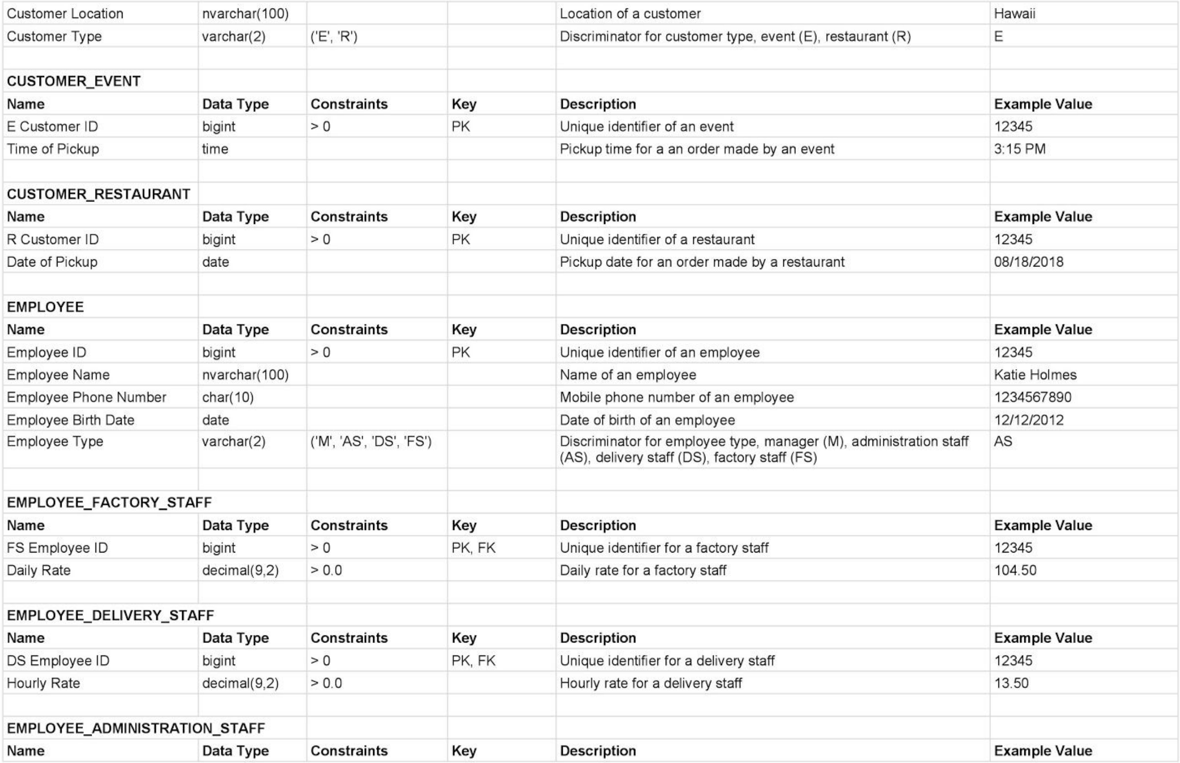
Name	Data Type	Constraints	Key	Description	Example Value
Item Type ID	bigint	> 0	PK	Unique identifier for an item type	12345
Types of Item	nvarchar(100)			Different types of items for order	Glass Bottle

ORDER

Name	Data Type	Constraints	Key	Description	Example Value
Order ID	bigint	> 0	PK	Unique identifier for an order	12345
Order Date	date			Date and time of an order	11/21/2018 5:25 PM
Ordered Quantity	int	> 0		Quantity of containers ordered	10
CustomerID	bigint		FK	Customer associated with an order; unique identifier for a customer	12345

LOAN

Name	Data Type	Constraints	Key	Description	Example Value
Loan ID	bigint	> 0	PK	Container that has been ordered; surrogate identifier for an ordered container with the unit cost	12345
Unit Cost	decimal(9,2)	> 0.0		Cost per unit of an item	20.00
Item Type ID	bigint	> 0	FK	Item type associated with an order and a unit cost; unique identifier for an item type	12345
Order ID	bigint	> 0	FK	Order associated with a container and a unit cost; unique identifier for an order	12345



Customer Location	nvarchar(100)			Location of a customer	Hawaii
Customer Type	varchar(2)	('E', 'R')		Discriminator for customer type, event (E), restaurant (R)	E

CUSTOMER_EVENT

Name	Data Type	Constraints	Key	Description	Example Value
E Customer ID	bigint	> 0	PK	Unique identifier of an event	12345
Time of Pickup	time			Pickup time for a an order made by an event	3:15 PM

CUSTOMER_RESTAURANT

Name	Data Type	Constraints	Key	Description	Example Value
R Customer ID	bigint	> 0	PK	Unique identifier of a restaurant	12345
Date of Pickup	date			Pickup date for an order made by a restaurant	08/18/2018

EMPLOYEE

Name	Data Type	Constraints	Key	Description	Example Value
Employee ID	bigint	> 0	PK	Unique identifier of an employee	12345
Employee Name	nvarchar(100)			Name of an employee	Katie Holmes
Employee Phone Number	char(10)			Mobile phone number of an employee	1234567890
Employee Birth Date	date			Date of birth of an employee	12/12/2012
Employee Type	varchar(2)	('M', 'AS', 'DS', 'FS')		Discriminator for employee type, manager (M), administration staff (AS), delivery staff (DS), factory staff (FS)	AS

EMPLOYEE_FACTORY_STAFF

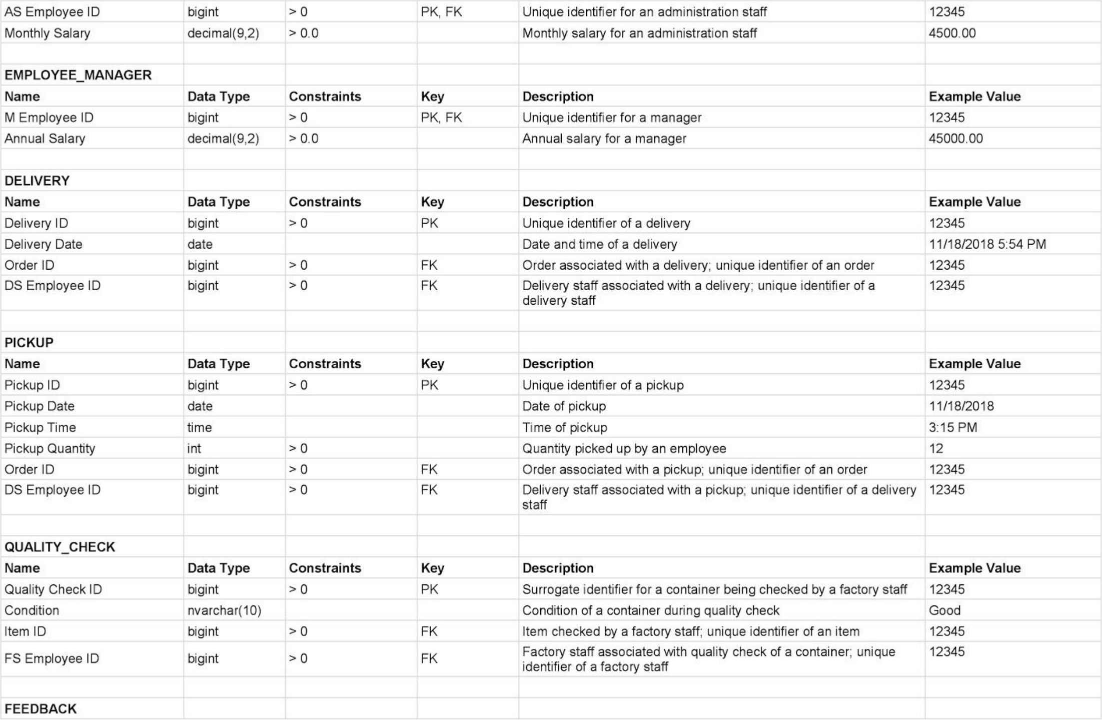
Name	Data Type	Constraints	Key	Description	Example Value
FS Employee ID	bigint	> 0	PK, FK	Unique identifier for a factory staff	12345
Daily Rate	decimal(9,2)	> 0.0		Daily rate for a factory staff	104.50

EMPLOYEE_DELIVERY_STAFF

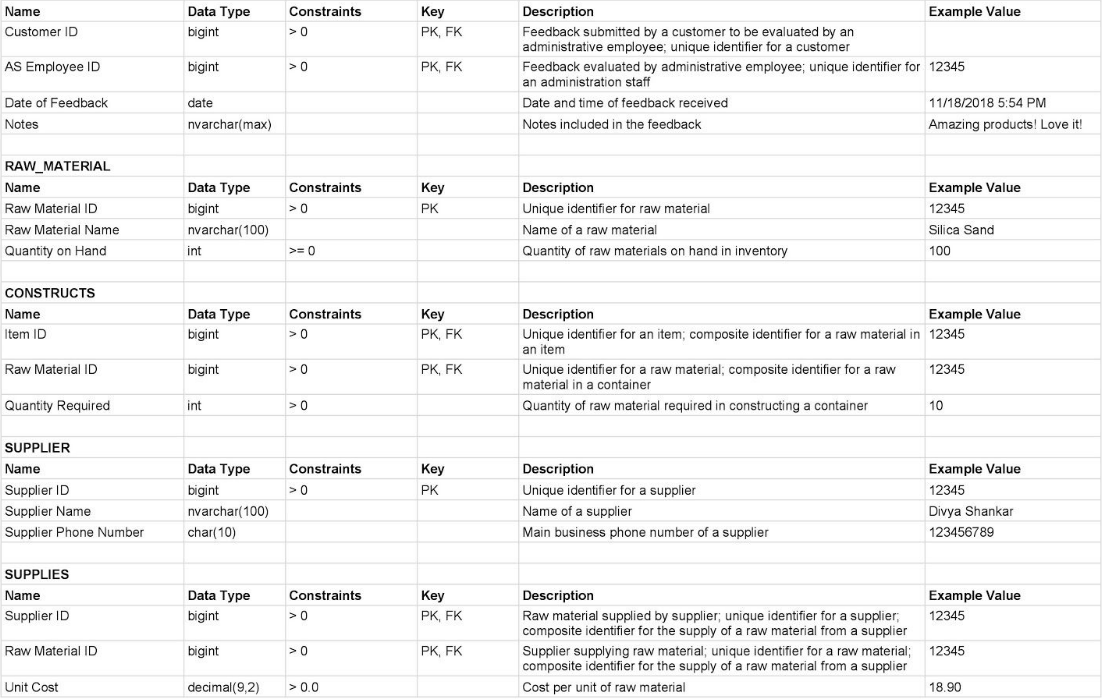
Name	Data Type	Constraints	Key	Description	Example Value
DS Employee ID	bigint	> 0	PK, FK	Unique identifier for a delivery staff	12345
Hourly Rate	decimal(9,2)	> 0.0		Hourly rate for a delivery staff	13.50

EMPLOYEE_ADMINISTRATION_STAFF

Name	Data Type	Constraints	Key	Description	Example Value



AS Employee ID	bigint	> 0	PK, FK	Unique identifier for an administration staff	12345
Monthly Salary	decimal(9,2)	> 0.0		Monthly salary for an administration staff	4500.00
EMPLOYEE_MANAGER					
Name	**Data Type**	**Constraints**	**Key**	**Description**	**Example Value**
M Employee ID	bigint	> 0	PK, FK	Unique identifier for a manager	12345
Annual Salary	decimal(9,2)	> 0.0		Annual salary for a manager	45000.00
DELIVERY					
Name	**Data Type**	**Constraints**	**Key**	**Description**	**Example Value**
Delivery ID	bigint	> 0	PK	Unique identifier of a delivery	12345
Delivery Date	date			Date and time of a delivery	11/18/2018 5:54 PM
Order ID	bigint	> 0	FK	Order associated with a delivery; unique identifier of an order	12345
DS Employee ID	bigint	> 0	FK	Delivery staff associated with a delivery; unique identifier of a delivery staff	12345
PICKUP					
Name	**Data Type**	**Constraints**	**Key**	**Description**	**Example Value**
Pickup ID	bigint	> 0	PK	Unique identifier of a pickup	12345
Pickup Date	date			Date of pickup	11/18/2018
Pickup Time	time			Time of pickup	3:15 PM
Pickup Quantity	int	> 0		Quantity picked up by an employee	12
Order ID	bigint	> 0	FK	Order associated with a pickup; unique identifier of an order	12345
DS Employee ID	bigint	> 0	FK	Delivery staff associated with a pickup; unique identifier of a delivery staff	12345
QUALITY_CHECK					
Name	**Data Type**	**Constraints**	**Key**	**Description**	**Example Value**
Quality Check ID	bigint	> 0	PK	Surrogate identifier for a container being checked by a factory staff	12345
Condition	nvarchar(10)			Condition of a container during quality check	Good
Item ID	bigint	> 0	FK	Item checked by a factory staff; unique identifier of an item	12345
FS Employee ID	bigint	> 0	FK	Factory staff associated with quality check of a container; unique identifier of a factory staff	12345
FEEDBACK					



Name	**Data Type**	**Constraints**	**Key**	**Description**	**Example Value**
Customer ID	bigint	> 0	PK, FK	Feedback submitted by a customer to be evaluated by an administrative employee; unique identifier for a customer	
AS Employee ID	bigint	> 0	PK, FK	Feedback evaluated by administrative employee; unique identifier for an administration staff	12345
Date of Feedback	date			Date and time of feedback received	11/18/2018 5:54 PM
Notes	nvarchar(max)			Notes included in the feedback	Amazing products! Love it!
RAW_MATERIAL					
Name	**Data Type**	**Constraints**	**Key**	**Description**	**Example Value**
Raw Material ID	bigint	> 0	PK	Unique identifier for raw material	12345
Raw Material Name	nvarchar(100)			Name of a raw material	Silica Sand
Quantity on Hand	int	>= 0		Quantity of raw materials on hand in inventory	100
CONSTRUCTS					
Name	**Data Type**	**Constraints**	**Key**	**Description**	**Example Value**
Item ID	bigint	> 0	PK, FK	Unique identifier for an item; composite identifier for a raw material in an item	12345
Raw Material ID	bigint	> 0	PK, FK	Unique identifier for a raw material; composite identifier for a raw material in a container	12345
Quantity Required	int	> 0		Quantity of raw material required in constructing a container	10
SUPPLIER					
Name	**Data Type**	**Constraints**	**Key**	**Description**	**Example Value**
Supplier ID	bigint	> 0	PK	Unique identifier for a supplier	12345
Supplier Name	nvarchar(100)			Name of a supplier	Divya Shankar
Supplier Phone Number	char(10)			Main business phone number of a supplier	123456789
SUPPLIES					
Name	**Data Type**	**Constraints**	**Key**	**Description**	**Example Value**
Supplier ID	bigint	> 0	PK, FK	Raw material supplied by supplier; unique identifier for a supplier; composite identifier for the supply of a raw material from a supplier	12345
Raw Material ID	bigint	> 0	PK, FK	Supplier supplying raw material; unique identifier for a raw material; composite identifier for the supply of a raw material from a supplier	12345
Unit Cost	decimal(9,2)	> 0.0		Cost per unit of raw material	18.90

SQL Statements

Table Creation

1. Item_T

```
CREATE TABLE Item_T
(ItemID int not null CHECK (ItemID > 0),
ItemStatus nvarchar(10),
ItemTypeID int not null,
CONSTRAINT Item_PK PRIMARY KEY (ItemID),
CONSTRAINT Item_Type_FK FOREIGN KEY (ItemTypeID) REFERENCES Item_Type_T
(ItemTypeID))
```

2. Item_Type_T

```
CREATE TABLE ITEM_TYPE_T
(ItemTypeID int not null CHECK (ItemTypeID > 0),
TypesofItem nvarchar(100)
CONSTRAINT Item_Type_PK PRIMARY KEY (ItemTypeID))
```

3. Customer_T

```
CREATE TABLE Customer_T
(CustomerID int not null CHECK (CustomerID > 0),
CustomerName nvarchar(100),
CustomerLocation nvarchar(100),
CustomerType varchar(2) CHECK (CustomerType IN ('E', 'R')) not null
CONSTRAINT Customer_PK PRIMARY KEY (CustomerID))
```

4. Customer_Event_T

```
CREATE TABLE Customer_Event_T
(ECustomerID int not null CHECK (ECustomerID > 0),
TimeofPickup time
CONSTRAINT Customer_Event_PK PRIMARY KEY (ECustomerID)
CONSTRAINT Customer_Event_FK FOREIGN KEY (ECustomerID) REFERENCES Customer_T
(CustomerID))
```

5. Customer_Restaurant_T

```
CREATE TABLE Customer_Restaurant_T
(RCustomerID int not null CHECK (RCustomerID > 0),
DateofPickup date
CONSTRAINT Customer_Restaurant_PK PRIMARY KEY (RCustomerID)
CONSTRAINT Customer_Restaurant_FK FOREIGN KEY (RCustomerID) REFERENCES Customer_T
(CustomerID))
```

6. Order_T

```
CREATE TABLE Order_T
(OrderID int not null CHECK (OrderID > 0),
OrderDate date default GETDATE(),
OrderedQuantity int not null CHECK (OrderedQuantity > 0),
CustomerID int
CONSTRAINT Order_PK PRIMARY KEY (OrderID),
CONSTRAINT Customer_FK FOREIGN KEY (CustomerID) REFERENCES Customer_T (CustomerID))
```

7. Loan_T

```
CREATE TABLE Loan_T
(LoanID int not null CHECK (LoanID > 0),
UnitCost decimal(9,2) not null CHECK (UnitCost > 0.0),
ItemTypeID int not null,
OrderID int not null
CONSTRAINT Loan_PK PRIMARY KEY (LoanID),
CONSTRAINT Item_Type_FK1 FOREIGN KEY (ItemTypeID) REFERENCES Item_Type_T
(ItemTypeID),
CONSTRAINT Order_FK FOREIGN KEY (OrderID) REFERENCES Order_T (OrderID))
```

8. Employee_T

```
CREATE TABLE Employee_T
(EmployeeID int not null CHECK (EmployeeID > 0),
EmployeeName nvarchar(100),
EmployeePhoneNumber char(10),
EmployeeBirthDate date,
EmployeeType varchar(2) CHECK (EmployeeType IN ('M', 'AS', 'DS', 'FS')) not null
CONSTRAINT Employee_PK PRIMARY KEY (EmployeeID))
```

9. Employee_Factory_Staff_T

```
CREATE TABLE Employee_Factory_Staff_T
(FSEmployeeID int not null CHECK (FSEmployeeID > 0),
DailyRate decimal(9,2) not null CHECK (DailyRate > 0.0)
CONSTRAINT Employee_Factory_Staff_PK PRIMARY KEY (FSEmployeeID)
CONSTRAINT Employee_Factory_Staff_FK FOREIGN KEY (FSEmployeeID) REFERENCES
Employee_T (EmployeeID))
```

10. Employee_Delivery_Staff_T

```
CREATE TABLE Employee_Delivery_Staff_T
(DSEmployeeID int not null CHECK (DSEmployeeID > 0),
HourlyRate decimal(9,2) not null CHECK (HourlyRate > 0.0)
CONSTRAINT Employee_Delivery_Staff_PK PRIMARY KEY (DSEmployeeID)
CONSTRAINT Employee_Delivery_Staff_FK FOREIGN KEY (DSEmployeeID) REFERENCES
Employee_T (EmployeeID))
```

11. Employee_Administration_Staff_T

```
CREATE TABLE Employee_Administration_Staff_T
(ASEmployeeID int not null CHECK (ASEmployeeID > 0),
MonthlySalary decimal(9,2) not null CHECK (MonthlySalary > 0.0)
CONSTRAINT Employee_Administration_Staff_PK PRIMARY KEY (ASEmployeeID)
CONSTRAINT Employee_Administration_Staff_FK FOREIGN KEY (ASEmployeeID) REFERENCES
Employee_T (EmployeeID))
```

12. Employee_Manager_T

```
CREATE TABLE Employee_Manager_T
(MEmployeeID int not null CHECK (MEmployeeID > 0),
AnnualSalary decimal(9,2) not null CHECK (AnnualSalary > 0.0)
CONSTRAINT Employee_Manager_PK PRIMARY KEY (MEmployeeID)
CONSTRAINT Employee_Manager_FK FOREIGN KEY (MEmployeeID) REFERENCES Employee_T
(EmployeeID))
```

13. Delivery_T

```
CREATE TABLE Delivery_T
(DeliveryID int not null CHECK (DeliveryID > 0),
DeliveryDate date,
OrderID int not null CHECK (OrderID > 0),
DSEmployeeID int not null CHECK (DSEmployeeID > 0)
CONSTRAINT Delivery_PK PRIMARY KEY (DeliveryID)
CONSTRAINT Order_FK1 FOREIGN KEY (OrderID) REFERENCES Order_T (OrderID),
CONSTRAINT Employee_Delivery_Staff_FK1 FOREIGN KEY (DSEmployeeID) REFERENCES
Employee_Delivery_Staff_T (DSEmployeeID))
```

14. Pickup_T

```
CREATE TABLE Pickup_T
(PickupID int not null CHECK (PickupID > 0),
PickupDate date,
PickupTime time,
PickupQuantity int not null,
OrderID int not null CHECK (OrderID > 0),
DSEmployeeID int not null CHECK (DSEmployeeID > 0)
CONSTRAINT Pickup_PK PRIMARY KEY (PickupID)
CONSTRAINT Order_FK2 FOREIGN KEY (OrderID) REFERENCES Order_T (OrderID),
CONSTRAINT Employee_Delivery_Staff_FK2 FOREIGN KEY (DSEmployeeID) REFERENCES
Employee_Delivery_Staff_T (DSEmployeeID))
```

15. Quality_Check_T

```
CREATE TABLE Quality_Check_T
(QualityCheckID int not null CHECK (QualityCheckID > 0),
Condition nvarchar(10),
ItemID int not null CHECK (ItemID > 0),
FSEmployeeID int not null CHECK (FSEmployeeID > 0)
CONSTRAINT Quality_Check_PK PRIMARY KEY (QualityCheckID)
CONSTRAINT Item_FK1 FOREIGN KEY (ItemID) REFERENCES Item_T (ItemID),
CONSTRAINT Employee_Factory_Staff_FK1 FOREIGN KEY (FSEmployeeID) REFERENCES
Employee_Factory_Staff_T (FSEmployeeID))
```

16. Feedback_T

```
CREATE TABLE Feedback_T
(CustomerID int not null CHECK (CustomerID > 0),
ASEmployeeID int not null CHECK (ASEmployeeID > 0),
DateofFeedback date,
Notes nvarchar(max)
CONSTRAINT Feedback_PK PRIMARY KEY (CustomerID, ASEmployeeID)
CONSTRAINT Customer_FK1 FOREIGN KEY (CustomerID) REFERENCES Customer_T
(CustomerID),
CONSTRAINT Employee_Administration_Staff_FK1 FOREIGN KEY (ASEmployeeID) REFERENCES
Employee_Administration_Staff_T (ASEmployeeID))
```

17. Raw_Material_T

```
CREATE TABLE Raw_Material_T
(RawMaterialID int not null CHECK (RawMaterialID > 0),
RawMaterialName nvarchar(100),
QuantityonHand int not null CHECK (QuantityonHand >= 0)
CONSTRAINT Raw_Material_PK PRIMARY KEY (RawMaterialID))
```

18. Constructs_T

```
CREATE TABLE Constructs_T
(ItemID int not null CHECK (ItemID > 0),
RawMaterialID int not null CHECK (RawMaterialID > 0),
QuantityRequired int not null CHECK (QuantityRequired > 0)
CONSTRAINT Constructs_PK PRIMARY KEY (ItemID, RawMaterialID)
CONSTRAINT Item_FK2 FOREIGN KEY (ItemID) REFERENCES Item_T (ItemID),
CONSTRAINT Raw_Material_FK FOREIGN KEY (RawMaterialID) REFERENCES Raw_Material_T
(RawMaterialID))
```

19. Supplier_T

```
CREATE TABLE Supplier_T
(SupplierID int not null CHECK (SupplierID > 0),
SupplierName nvarchar(100),
SupplierPhoneNumber char(10)
CONSTRAINT Supplier_PK PRIMARY KEY (SupplierID))
```

20. Supplies_T

```
CREATE TABLE Supplies_T
(SupplierID int not null CHECK (SupplierID > 0),
RawMaterialID int not null CHECK (RawMaterialID > 0),
UnitCost decimal(9,2) not null CHECK (UnitCost > 0.0)
CONSTRAINT Supplies_PK PRIMARY KEY (SupplierID, RawMaterialID)
CONSTRAINT Supplier_FK FOREIGN KEY (SupplierID) REFERENCES Supplier_T (SupplierID),
CONSTRAINT Raw_Material_FK2 FOREIGN KEY (RawMaterialID) REFERENCES Raw_Material_T
(RawMaterialID))
```

Materialized Views & Procedures

1. Delivery Staff

(Re)^3 has customized views for delivery and pickup details for the delivery staff. The delivery details view provide an overview of the customer name, customer location, type of items, quantity ordered, and delivery date. The pick-up details view provides specific pick up date and time, in addition to the details captured by the delivery details.

The reason why (Re)^3 has separate view for delivery details and pick up details is because, the pick-up date and time are requested by a customer, and not all delivery have a pickup schedule associated with it. To prevent the view from having too many null values, the company has decided to divide them.

VIEW 1: Delivery Details

The delivery details view would help (Re)^3 in the following ways:

- Ensure no mixup of items delivered, or missed delivery date.
- Prevent erroneous delivery of items to customers.
- Ensure fulfillment of a delivery associated with an order.

```
CREATE TABLE DeliveryDetails_View
(CustomerID int not null,
CustomerName nvarchar(100),
CustomerLocation nvarchar(100),
OrderID int not null,
OrderedQuantity int,
ItemTypeID int not null,
TypesofItem nvarchar(100),
DeliveryID int not null,
DeliveryDate date)
```

```
CREATE PROCEDURE RefreshDeliveryDetails_View as
delete from DeliveryDetails_View
insert into DeliveryDetails_View
select customer_t.customerid, customer_t.customername, customer_t.customerlocation,
order_t.orderid,
order_t.orderedquantity, item_type_t.itemtypeid, item_type_t.typesofitem,
delivery_t.deliveryid, delivery_t.deliverydate
from customer_t, order_t, delivery_t, item_type_t, loan_t
where customer_t.customerid = order_t.customerid
and order_t.orderid = loan_t.orderid
and loan_t.itemtypeid = item_type_t.itemtypeid
and order_t.orderid = delivery_t.orderid
```

VIEW 2: Pickup Details

The Pickup details view would help (Re)^{3} in the following ways:

- Prevent delays in pickup schedules.
- The company can track the differences between the delivered quantities and pick up quantities.

```
CREATE TABLE PickupDetails_View
(CustomerID int not null,
CustomerName nvarchar(100),
CustomerLocation nvarchar(100),
PickupID int not null,
PickupDate date,
PickupTime time,
PickupQuantity int)
```

```
CREATE PROCEDURE RefreshPickupDetails_View as
delete from PickupDetails_View
insert into PickupDetails_View
select customer_t.customerid, customer_t.customername, customer_t.customerlocation,
pickup_t.pickupid, pickup_t.pickupdate, pickup_t.pickuptime, pickup_t.pickupquantity
from customer_t, pickup_t, order_t
where customer_t.customerid = order_t.customerid
and order_t.orderid = pickup_t.orderid
```

2. Administration Staff

The administration staff at (Re)^3 have customized views for order details and feedback details. The order details are specifically for the accounting department and the feedback details are for the customer service department.

VIEW 1: Order Details

The Order details view would help (Re)^3 in the following ways:

- Track the profit of (Re)^3.
- Track the types of items that are generating most profit for the company.
- With advanced machine learning techniques, (Re)^3 can track companies that order frequently to generate insights about their orders, and reward them with discounts associated with their orders.

```
CREATE TABLE OrderDetails_View
(CustomerID int not null,
CustomerName nvarchar(100),
OrderID int not null,
OrderDate date,
OrderedQuantity int,
LoanID int not null,
UnitCOst decimal(9,2),
TotalAmount money,
ItemTypeID int not null,
TypesofItem nvarchar(100))
```

```
CREATE PROCEDURE RefreshOrderDetails_View as
delete from OrderDetails_View
insert into OrderDetails_View
select customer_t.customerid, customer_t.customername, order_t.orderid, order_t.orderdate,
order_t.orderedquantity, loan_t.loanid, loan_t.unitcost,
(order_t.orderedquantity * loan_t.unitcost),
item_type_t.itemtypeid, item_type_t.typesofitem
from customer_t, order_t, loan_t, item_type_t
where customer_t.customerid = order_t.orderid
and order_t.orderid = loan_t.orderid
and loan_t.itemtypeid = item_type_t.itemtypeid
```

VIEW 2: Feedback Details

The feedback details view would help (Re)^3 in the following ways:

- Allow the staff to track the feedback date, feedback notes, customer and order ID associated with a feedback.
- The administration staff will use this feedback to improve pickup/delivery service, manufacturing and quality check for the items provided.

```
CREATE TABLE FeedbackDetails_View
(CustomerID int not null,
CustomerName nvarchar(100),
OrderID int not null,
OrderDate date,
Quantity,
ItemTypeID int not null,
TypesofItem nvarchar(100),
DateofFeedback date,
Notes nvarchar(max))
```

```
CREATE PROCEDURE RefreshFeedbackDetails_View as
delete from FeedbackDetails_View
insert into FeedbackDetails_View
select customer_t.customerid, customer_t.customername, order_t.orderid, order_t.orderdate,
count(distinct(loan_t.itemtypeid)) as Quantity, item_type_t.itemtypeid,
item_type_t.typesofitem,
feedback_t.dateoffeedback, feedback_t.notes
from customer_t, item_t, order_t, item_type_t, feedback_t, loan_t
where order_t.customerid = customer_t.customerid
and loan_t.itemtypeid = item_type_t.itemtypeid
and feedback_t.customerid = customer_t.customerid
group by customer_t.customerid, customer_t.customername, order_t.orderid,
order_t.orderdate,
item_type_t.itemtypeid, item_type_t.typesofitem,
feedback_t.dateoffeedback, feedback_t.notes
```

3. Factory Staff

At (Re)^3, the factory staff have customized views for quality check and raw material supply. These views help in the business operations pertaining to in-house manufacturing of items to be rented, and high quality cleaning of the items that are supplied to a variety of customers.

VIEW 1: Quality Check

The quality check view would help (Re)^3 in the following ways:

- The staff have an overview of the quality of an item, item status, type of item, and condition of the item supplied.
- In the case of a return with a damaged item, (Re)^3 can track the customer associated with the order and take necessary action such as notifying/blacklisting them.

```
CREATE TABLE QualityCheck_View
(ItemID int not null,
ItemStatus nvarchar(10),
Condition nvarchar(10),
ItemTypeID int not null,
TypesofItem nvarchar(100),
CustomerID int not null,
CustomerName nvarchar(100))
```

```
CREATE PROCEDURE RefreshQualityCheck_View as
delete from QualityCheck_View
insert into QualityCheck_View
select item_t.itemid, item_t.itemstatus, quality_check_t.condition,
item_type_t.itemtypeid, item_type_t.typesofitem,
customer_t.customerid, customer_t.customername
from item_t, quality_check_t, customer_t, item_type_t, loan_t, order_t
where item_t.itemid = quality_check_t.itemid
and item_t.itemtypeid = loan_t.itemtypeid
and item_type_t.itemtypeid = loan_t.itemtypeid
and loan_t.orderid = order_t.orderid
and order_t.customerid = customer_t.customerid
```

VIEW 2: Supply Details

The supply details view would help (Re)^3 in the following ways:

- The factory staff at the manufacturing department have an overview of the demand and supply of items with the raw materials supply view.
- The factory staff can track the demand supply ratio for items to be supplied. In the event of high demand and low supply, the staff place a manufacture request for items low on stock.
- The factory staff can keep track of raw materials used in manufacturing, and cleaning along with the quantity on hand. In the event of low stock of a raw material, the staff place a supply request from the respective supplier.

```
CREATE TABLE SupplyDetails_View
(ItemID int not null,
ItemStatus nvarchar(10),
ItemTypeID int not null,
TypesofItem nvarchar(100),
OrderedQuantity int,
PickupQuantity int,
RawMaterialID int not null,
QuantityonHand int,
QuantityRequired int)

CREATE PROCEDURE RefreshSupplyDetails_View as
delete from SupplyDetails_View
insert into SupplyDetails_View
select item_t.itemid, item_t.itemstatus, item_type_t.itemtypeid,
item_type_t.typesofitem, order_t.orderedquantity, pickup_t.pickupquantity,
raw_material_t.rawmaterialid, raw_material_t.quantityonhand, constructs_t.quantityrequired
from item_t, item_type_t, order_t, pickup_t, raw_material_t, constructs_t, loan_t
where item_t.itemtypeid = item_type_t.itemtypeid
and loan_t.itemtypeid = item_type_t.itemtypeid
and loan_t.orderid = order_t.orderid
and pickup_t.orderid = order_t.orderid
and constructs_t.itemid = item_t.itemid
and constructs_t.rawmaterialid = raw_material_t.rawmaterialid
```

Database Triggers

Trigger 1: Price Log

- The Price log trigger would help to keep a record of all the changes made to the unit cost for an item be rented out.
- This trigger will record information on the old unit cost, new unit cost and the date of update.
- This trigger would help the accounting department by maintaining consistency of pricing data, identifying malicious attempts to alter the price of an item and facilitate audit operations.

```
CREATE TABLE PriceUpdates_Log
(ItemTypeID int,
OldPrice float,
NewPrice float,
UpdateDate datetime)

CREATE TRIGGER PriceUpdate on Loan_T
for update
as
if update(unitcost)
begin
insert into PriceUpdates_Log (itemtypeid, oldprice,
newprice, updatedate)
select inserted.itemtypeid,
deleted.unitcost, inserted.unitcost, GETDATE()
from inserted, deleted where inserted.itemtypeid = deleted.itemtypeid
end
```

Trigger 2: Delivery Log

- The Delivery log trigger would help to keep a record of the delivery associated with an order.
- This trigger will record information on the expected delivery date, delivered date, delivery staff servicing the delivery, the status of delivery and updated date if delivered.
- This trigger would ensure fulfillment of delivery services, avoid mix up of deliveries and erroneous delivery of orders.

```
CREATE TABLE DeliveryUpdates_Log
(DeliveryID int not null,
OrderIDold int,
OrderIDnew int,
DSEmployeeIDold int,
DSEmployeeIDnew int,
DeliveryDateold date,
DeliveryDatenew date,
Updated char(1),
Deleted char(1),
UpdateDate datetime)
```

```
CREATE TRIGGER DeliveryUpdate on Delivery_T
for update, delete
as
declare @deliveryid int, @orderidold int, @orderidnew int,
@DSEmployeeold int, @DSEmployeenew int, @deliverydateold date, @deliverydatenew date

if (exists (select * from deleted) and exists (select * from inserted))
begin
insert into DeliveryUpdates_Log (deliveryid, orderidold, orderidnew,
dsemployeeidold, dsemployeeidnew, deliverydateold, deliverydatenew, updated, deleted,
updatedate)
select inserted.deliveryid, deleted.orderid, inserted.orderid,
deleted.dsemployeeid, inserted.dsemployeeid, deleted.deliverydate, inserted.deliverydate, 'Y',
'N', getdate()
from inserted, deleted
where inserted.deliveryid = deleted.deliveryid
end

if (exists (select * from deleted) and not (exists (select * from inserted)))
begin
insert into DeliveryUpdates_Log (deliveryid, orderidold, orderidnew,
dsemployeeidold, dsemployeeidnew, deliverydateold, deliverydatenew, updated, deleted,
updatedate)
select deleted.deliveryid, deleted.orderid, null, deleted.dsemployeeid, null,
deleted.deliverydate, null, 'N', 'Y', getdate() from deleted
end
```