**Apparel Store Database Management System**(A Retail Database Management Application)

DELOWER HOSSAIN

MET CS669 Spring 1, 2019 (Week 5)

**Boston University**

**Table Contents:**1. Project Overview ----------------------------------------------------------------------------------------------------

2. Iteration four Overview--------------------------------------------------------------------------------------------  
2.Software suite ---------------------------------------------------------------------------------------------------------

4. Use cases---------------------------------------------------------------------------------------------------------------

5. Structural database rules for Apparel store database------------------------------------------------------

6. ER modeling of ASDBMS-------------------------------------------------------------------------------------------

7. Composite of ERM for ASDBMS----------------------------------------------------------------------------------

9. Revised Generalization/Specialization Mapping - *EERD* ---------------------------------------------------

10. Adding datatype, attributes, and constraint in ERD-------------------------------------------------------

11. Normalized Physical EERD Creation---------------------------------------------------------------------------

12. Adding history table in Physical EERD Creation------------------------------------------------------------

13. Table and constraints creation in SQL developer----------------------------------------------------------  
14. Index Creation in SQL ---------------------------------------------------------------------------------------------

15. Data Insertion------------------------------------------------------------------------------------------------------

16. Reusable store procedures creation--------------------------------------------------------------------------

17. Questions & Queries

18. Summary & Reflecting--------------------------------------------------------------------------------------------

19. Reference -----------------------------------------------------------------------------------------------------------

Project Overview

A fascination to develop a database application named “Apparel Store Database Management System”. We assume that an apparel clothing brand having several stores across the country and everyday lots of transactions born from several sides such as customer, vendor, shipper, etc. Therefore, management needs to have informed about all aspects precisely. As concern, what is the sales volume for a month for specific store and how about for entire business? Which product is most popular? What is the last month total sales? Monthly order? Who are our top customers/vendors? Those are so essential aspects for the management. In fact, to make a business decision that key information needs to retrieve as timely fashion.

Development process applied database design principles, structural rules, created entity relational model and EERD. We reconstruct ERD with generalization-specialization rules, finally adding history table to keep tack the price. In addition, database has integrated some robust store procedures, essential indexing to make the query faster and fluent and constraints to make the field mandatory where needed.   
  
It is worth emphasizing to expose that the goal of this database is to serve a system for an apparel industry to Track and manage their Customers, Orders and Products.

Finally, The aspiration to build an apparel database system since I had my own export-oriented international apparel business. I might have picked up an innovative topic but for the sake of clarification and building database successfully, this topic would be more understandable aspects in this stage. As a result, I believe it I would be calmer to make happen the ASDMS development journey.

**Software suite considered:**

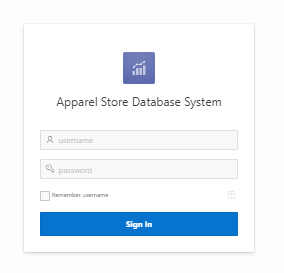
In order to build the database application, we use leading modern oracle RDBMS technology.   
The following software’s will be using throughout the entire term project.

* Oracle database 12c (Enterprise Edition)
* Enterprise Manager
* SQL developer
* SQL Plus/ Command Promote
* Oracle SQL developer data modeler/ Lucichart/draw.io
* Oracle apex 18.2 (Optional)

Use Case and Field

Creating user cases help us to identify the different capabilities (or features) we need to enable in our Database application design with some high-level dependencies, features/functionalities that user should capable to perform required tasks.   
  
Here are three user cases flows for the ASDMS application. The purpose of creating a user flow is to understand the steps a user will take as the complete a specific objective. Notice that there would be two end user one customer side another from organization admin, both role and privileges would be different.

1. **Signup/ Membership Use case :**



**Sign up for Account/membership**

**Login to application**

**Provide email, user name, Password, etc. for signup**

The Signup use case flows explained detail as below;

1. A user opens a Web browser and connects to the ASDMS;

2. The web application offers two options – existing users login and create a new account.  
Conversely, A Customer can be a member having present in a store to get offer and discount.

3. User choses to create a new account, a signup form offer to enter personal and contact information such as user name, email id, and password among other details.   
  
4. Once a user (customer or employee) have an account they would be able to track all activities.   
  
From database point of view, step three and four are related with database. The data those insert during signup form will be storages in database for future record such as name, password, even once user attempts to login by username and password, database crosscheck/retrieve the account info for that use.

Likewise, the way users enter the data for specific field for instance; Customer Last name, First Name, User Name, email, etc. This will also help to implement them by SQL script to generate information from raw data. Here is an example [link](https://apex.oracle.com/pls/apex/f?p=101071:LOGIN_DESKTOP:114315914315739:::::) of above user login form of this database app.

For this use case, however, the remarkable fields can be illustrate as below.

|  |  |  |
| --- | --- | --- |
| **Field** | **Storage Data** | **Needed As** |
| User ID | This filed store unique ref number as identification | To retrieve data using ID also make a difference for all user/customer |
| Account Name | The name of the user | To display or print name |
| First Name | The First Name who holds the account | Needed during communication |
| Last Name | Last Name about the account | Same as First Name |
| Customer Since | The data when account was created | To aware about history of a user/customer |
| Contact \_No | Customer/account holder's contact | Required to be connected |

1. **Place an Order – Use case**

**Payment & Get Invoice**

**Select item & Qty**

**Place order**

**Login Account**

In this use case, transaction might be in a store or could be online. Assume that a customer is a member of our store and purchasing in person then a sales representative accomplish the transaction on behalf of a customer and finally would be generated an Invoice.  
  
This use case play a significant role in our database. To accomplish the above procedures, there are several programming components related. However, concerning our database perspective, we may have order/Invoice entity, customer entity and product entity those perform together to accomplish the above job.   
  
  
To be precious, this use case ultimately generates an Invoice or we can say a sales receipt. Thus, we are having an entity from this use case named Invoice entity. The following significant field may have for the Invoice entity.

|  |  |  |
| --- | --- | --- |
| **Field** | **Storage Data** | **Needed As** |
| Invoice\_No | This indicates unique identification for each Invoice. | This is needful to make a difference for all Invoice |
| Store\_Code | The sore number where purchase has been made | To get store basis report |
| Invoice\_Date | This is the date of purchase | Date is essential for day by day transaction |
| Cus\_ID | The customer number who has purchased | Needed to trace who made the purchase |
| Emp\_ID | Store employee's identification | Wise to have about who conducted the sale |

In addition to above Invoice entity, there are several entities may have a relation with Invoice use case. For instance; Customer, Product, store and employee and InvoiceLine. We will discuss those entities in sequential section.

1. **Add product – Use case**

As mentioned, our database is focus mainly to track and manage customer, order, and product. It means that database end users are would be organization IT professional, Administrator-Manager, or developer in addition to regular customers. Hence, this is one of the use case, which belongs for sales manager/administrator. There is flow diagram about how this use case works;

**Admin/Manager Login**

**New product added in product table, DB**

**Fill out the product related info,**

**Press Add Product**

To execute the above use case operation we can use SQL INSERT INTO script/syntax under the add product button as logic. However, in database terminology we can develop a important entity from this use case which is product entity.   
The entity of product may have below attributes as shows in a dummy grid. Each field is relevant with the specific product instance, accept data based on data type and constraint, and transfer into product table of a database

**Product Table Example**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Product\_Id | Product\_Name | Product\_On\_Hand | Product\_Price | Category\_Id | Ven\_Code |
| 101 | Shirt | 120 | 25 | MENS | GAP |
| 102 | Ladies\_Jacket | 95 | 35 | LADIES | H&M |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

We examine the above product table, which apparently indicates how Apparel storage product table’s data will be storage in a tabular format in our database. Above all columns represents as attributes of product table. Once user add specific product in product table the data will be inserted including related product attributes increasing the rows (as shows here two product is available in product table.   
  
The following is the significant fields of a product table;

**Product Table**

|  |  |  |
| --- | --- | --- |
| **Field** | **Storage Data** | **Needed As** |
| Product\_Id | This indicates unique identification for Product | This is needful to make a difference for all Product |
| Product\_Name | This is detail name of the product | To get informed about the product |
| Product\_Price | The price of each product | Price is essential to calculate growth & report. |
| Product\_On\_Hand | Total number of product currently available | Needed to trace how many we have |
| Vendor\_Code | Supplier information | Wise to have about who provides specific product |
| Category\_Code | Each product will have under a specific group | Easy to organized |

Indeed, we will have numerous use cases for ASDB beside the above such as sales manager can generate a sales report for specific day, moth or year. HR or admin can review the employee works related information. Last year best selling product. Best selling store etc. In the next, we will explain business rules or structural business rule to obtain our precious entities and their relation in terms of those use cases.

**Structural Database Rules**To develop our structural database rules we go through with our each use cases.   
 **Signup/ Membership Use case:**

To accomplish signup/ to become a member the following steps require to be completing;

1. A user opens a Web browser and connects to the ASDMS;

2. The web application offers two options – existing users login and create a new account.  
Conversely, A Customer can be a member having present in a store to get offer and discount.

3. User choses to create a new account, a signup form offer to enter personal and contact information such as user name, email id, and password among other details.   
4. Once a user (customer or employee) have an account they are all set to accomplish specific job.

From database terminology, however, step three and four are related with our database. Since our focus is database, as a result we obtain one important entity from this use case, which can be treated as account entity. As there are no other entity, we just keep a record this entity with related attributes.

**Place an order use case:**

In this case, a user follow the below tasks flows to place an order.  
   
1. User have an access to account that can be online or store.  
2. Select the desire product   
3. Place and order   
4. Process payment and generates Invoice.

“Place and Order” this use case plays a significant role for the store, database, alone business rules. We may get numerous entities for this use case. As observed four significant entity possible to develop from this use case. For example; Product, Store, and Invoice, Invoice\_Line, Customer those are directly related with this use case. ASDBMS of course, tracks which products are purchased, and belongs to which customer, by which store. Indeed, that information are needed to produce an Invoice against a transaction.   
  
In structural database rule perspective, we are obtaining the following rules from this use case;   
  
**Rule A**

Each STORE process many orders/INVOICE

An Invoice/order is associated with one STORE

**Rule B**

Each customer may generate one or more INVOICEs

Each INVOICE is generated by one CUSTOMER

**Rule C**

An INVOICE/order/Sales contains one or more INVOICE LINE.

An INVOICE\_LINE/order\_LINE/Sales LINE is associated with a single INVOICE.

**Rule D**

Each INVOICE LINE References one PRODUCT

Each PRODUCT is referenced in one or more INVOICE LINEs

**Add product use case:**This use case belongs to an administrative part or could be IT professional and below is how they will interact with database.  
  
1. Login account

2. Fill out the product all attributes and fields  
3. Add product in a product table  
  
for this use case, we will have three entities as product, category and vendor. Apparel store database should have data about the supplier whose are suppling the specific products. In addition, to organise the product we may group as category. Here are rules we may develop among these entities.

A VENDOR supplies zero or one to many PRODUCT.  
Each PRODUCT is shipped zero to many VENDOR

One PRODUCT belongs to a CATEGORY

Each CATEGORY is associated zero to many PRODUCT.

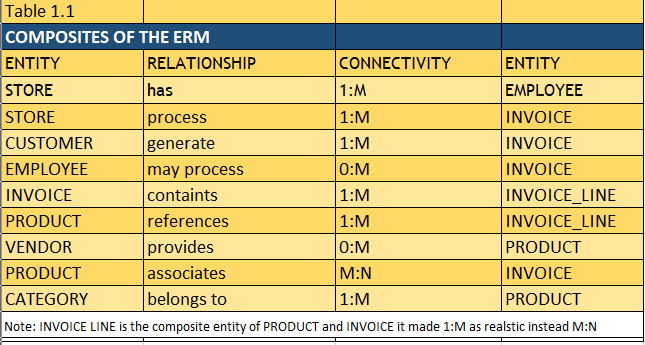
However, we obtained precious rules for our database, will discuses each components and their relationship in next sequential stage during ER modeling.

Based on uses case analysis, we can formed an initial decision to contemplate the following significant entities for our database. Therefore, the ASDBS sample database schema will consists of the following entities:

* **Customers**: stores customer’s data.
* **Products**: stores a list of apparel/ clothing products.
* **Category**: stores a group of product’s categories.
* **Invoice/Orders**: stores sales orders/Invoice placed by customers.
* **Invoice/Line**: stores sales invoice line items for each sales order/Invoice.
* **Vendor**: stores supplier’s data.
* **Employees**: stores all employee information as well as the organization structure such as who reports to whom.
* **Offices/store**: stores sales office data.

**Entity Relational Modeling**The purpose of entity relation model as shows a structure, a relation of data elements. This is a preeminent abstract way to describe database.

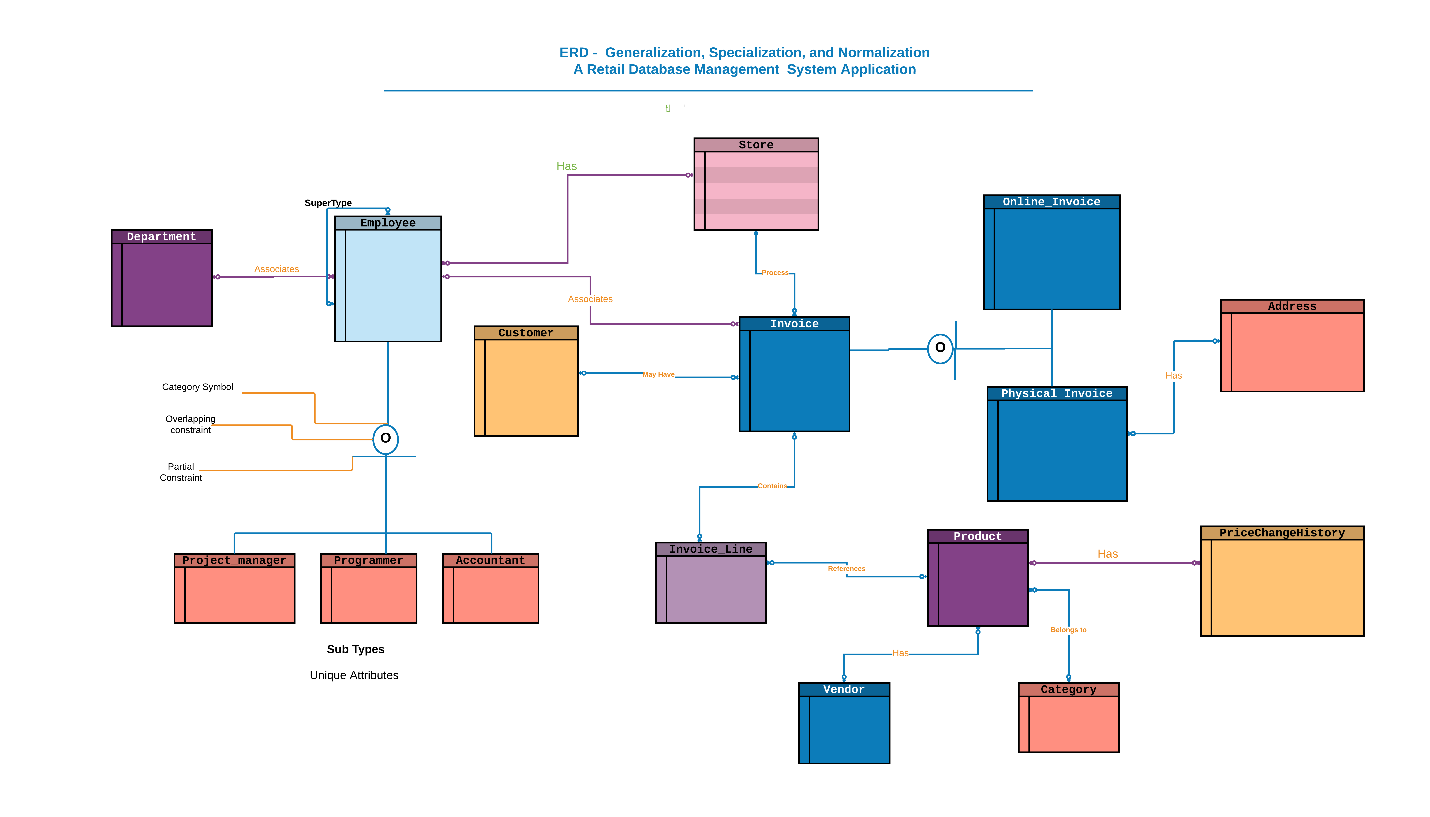
Since we have our entities and structural rules, in this phase, we will be developing ERM- an entity relational model, which plays a significant role for entire database life cycle. It is worthy to say that if structural rules do not reflect on ERM model entire database process would be insignificant.   
  
Here is a grid presentation of each entity and their relation for apparel store database.

****

# Extended ERD with Generalization/ Specialization Mapping

Based on our structural database rules (business rules) and EERM we can represent here our extended entity relation diagram (EERD).In this stage; we are developing our diagram by composite specialization-generalization relationships for ASDMS.

EER diagram can be we built with rules, using Crows foot since it is the most familiar to everyone and considerate as standard. The diagram represent each entity is in a relation by different lines, which are the Crow’s foot notation.



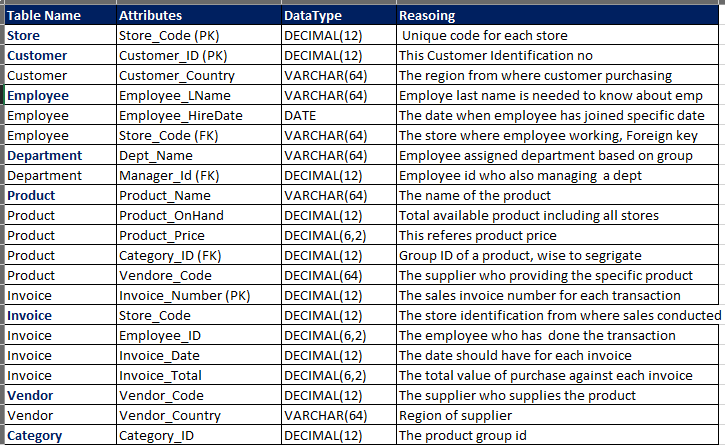
The above diagram is a conceptual EERD of apparel store DBMS.   
  
We can demonstrate the significant portion of the above diagram. As it shows Invoice entity is related at most four entities. Thus, we develop Invoice\_line entity as bridge entity since product and invoice having many to many relation (M:N) which is unrealistic should not have developed in RDBMS. Hence, we join both of them by another bridge entity named InvoiceLine and they are in a relation now as 1: M with Invoice Line instead M: N.

A specialization hierarchy (taxonomy) is a hierarchical structure that we use to shape our EERD.  
In our database perspective, we observed that employee entity could have several subtypes.  
As a result, there is a great scope to enhance an EERD for our Apparel store database.   
  
We defined employee is a super class which can be defined as parent class as well. In object-oriented perspective, parent class is a class, which holds all common attributes where subclass inherit those attributes from parent class. In our EERD diagram, it is important to appear there are several employees for instance; IT professional, Accountant all determine as employee though their role and skills are distinguish but having common attributes.

As we examine from the diagram, the type-subtype relationship on the lower left indicates that administrators and programmer or accountant are employees, that an employee can be both an administrator and a programmer or accountant, in addition that not all employees are either administrators or professors.

In this iteration, we have added price change history entity, which is related product table and we incorporated the physical DBMS accordingly.

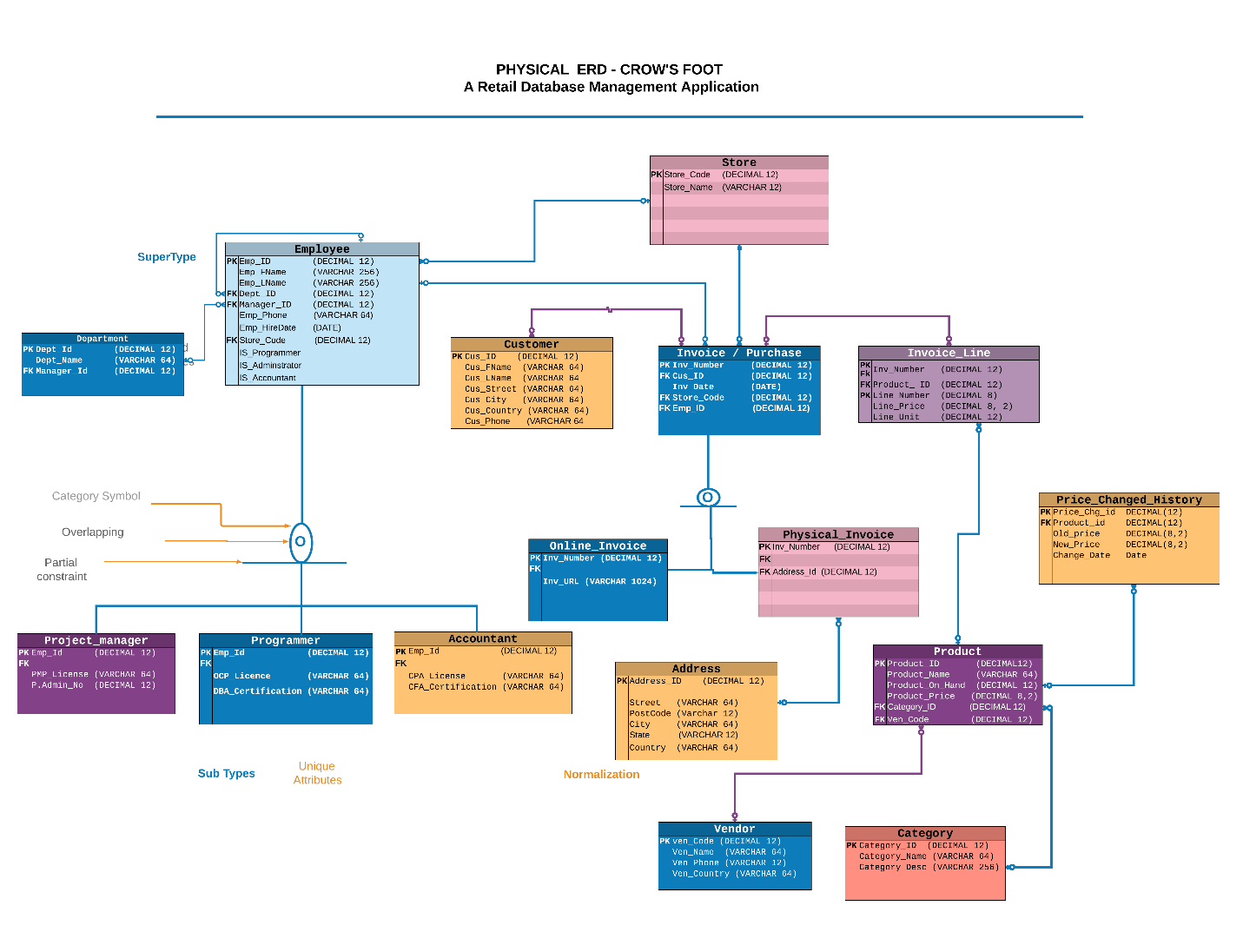
**Adding attribute on Physical ERD**

Now moving forward through the process of adding attributes for each entities and datatype. Here we have placed only significant attributes for entire db for demonstration purpose.   
  
  
  
  


In addition to above all attributes, we will demonstrate all columns/attributes in physical ERD.

**Normalized Physical ERD -ASDBMS**

Noticethat only one place where there is redundancy in my physical ERD, and that is with the address in formation in the Physical entity. If numerous purchases are made at that same by physically in a store, the address information will be duplication. Here is my EERD with the address information normalized.   
  
We have added history entity as price change history table to keep track all prices.

  
  
  
We obtained one additional entities after normalization, which is Address as we

Do not need to repeat the address every time during each purchase is conducted. We should reference the address instead.

Below are our additional structural database rules modified to reflect the new entities.

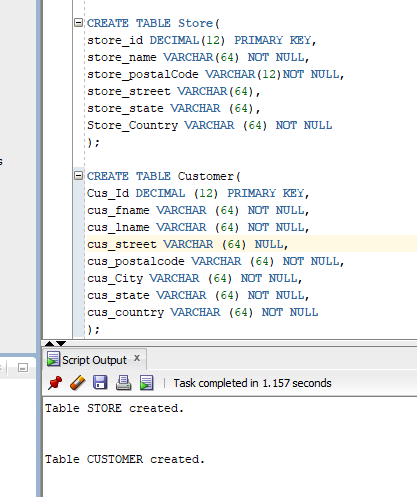
1. A purchase is an online purchase, a face‐to‐face purchase, both, or none of these.

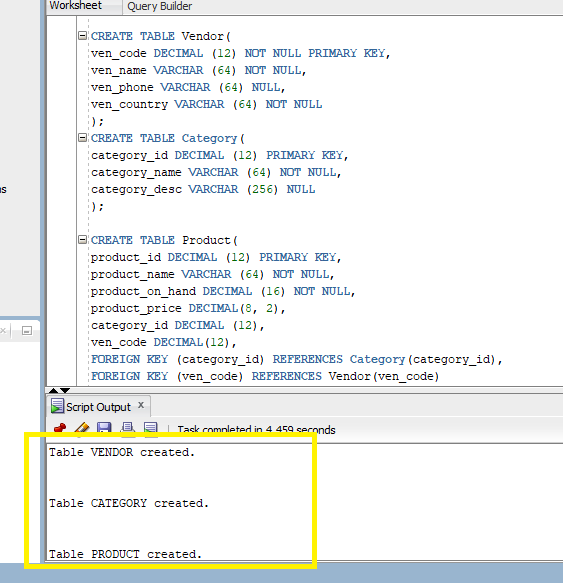
2. Each physical purchase is made at an address; each address is associated with one or more

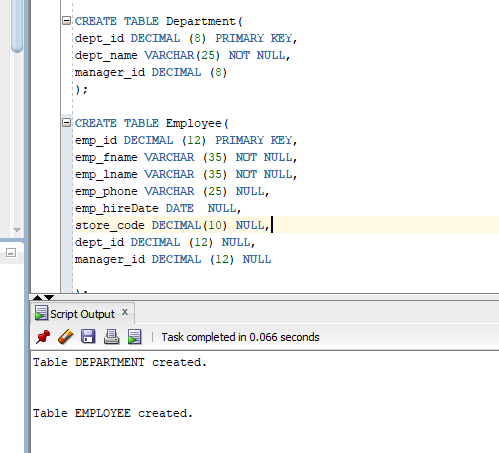
physical purchases.

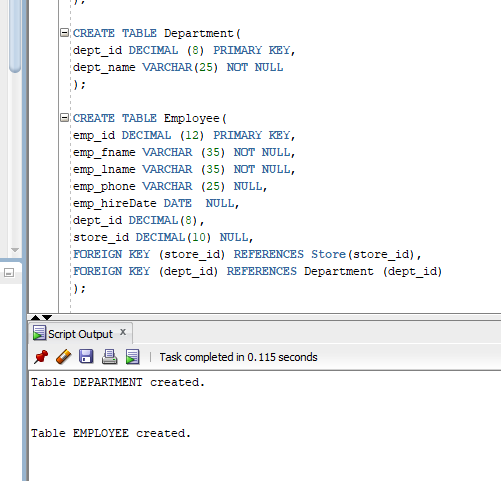
Since our modeling and physical ERD have been developed preciously, we moved ahead to create all tables with constraints and indexes as next drive.

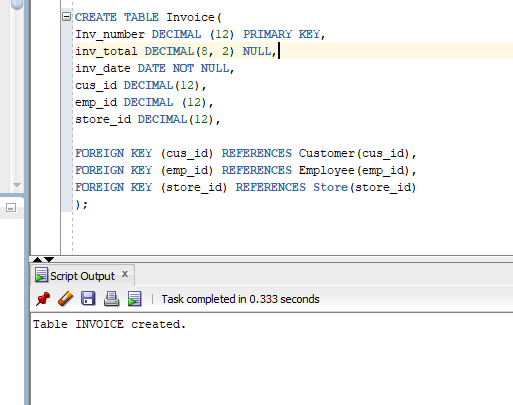
**Creating Tables from a DBMS Physical ERD**

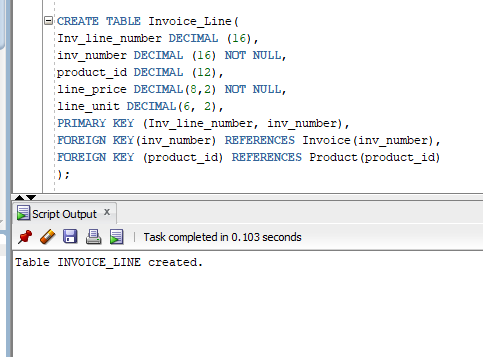


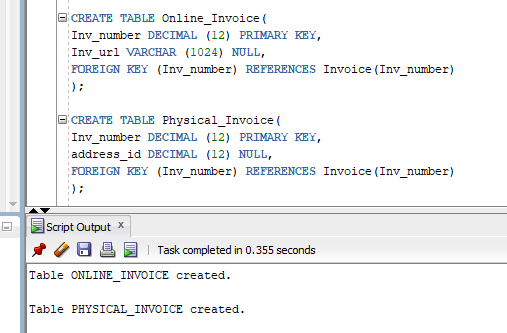


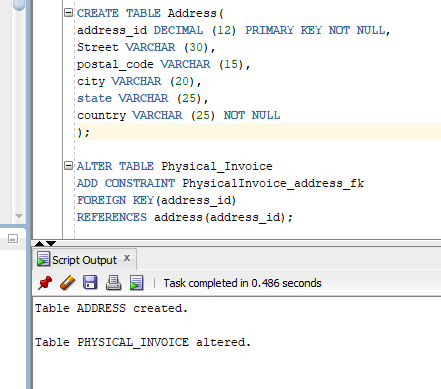


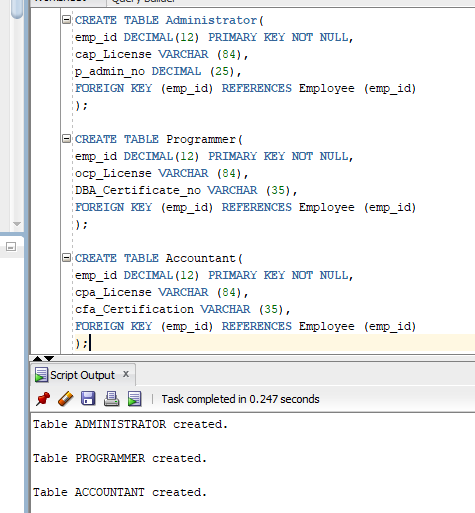


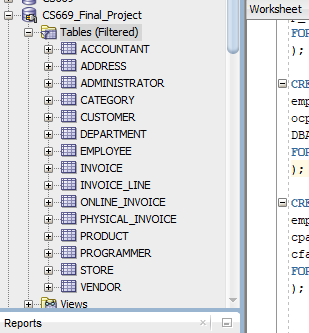




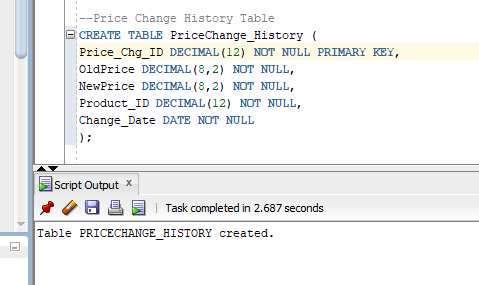








New Price Change History Table Creation



We have accomplished here 16 tables creation through SQL developer based on physical ERD.

Next, to speed up data retrieval and search process, we require to build some initial indexes those are essential.

**Creating Indexes in SQL for ASDBMS**

The following primary keys, which are already indexed by oracle DBMS, here is the list.  
  
**Primary Key Index**

**Store.Store\_id**

**Customer.Cus\_id**

**Vendor.Ven\_Code  
Category.Category\_id**

**Product.Product\_id**

**Department.Dept\_id**

**Employee.Emp\_id**

**Invoice.Inv\_Number**

**Invoice\_Line.Invoice\_Line\_Number**

**Online\_Invoice.Inv\_number**

**Physical\_Invoice.Inv\_Number**

**Address.Address\_id**

**Administrator.Emp\_id**

**Programmer.Emp\_id**

**Accountant.emp\_id**

As we know that foreign key indexing is optional but it is great to index foreign keys, we obtained below foreign keys to index for our database.

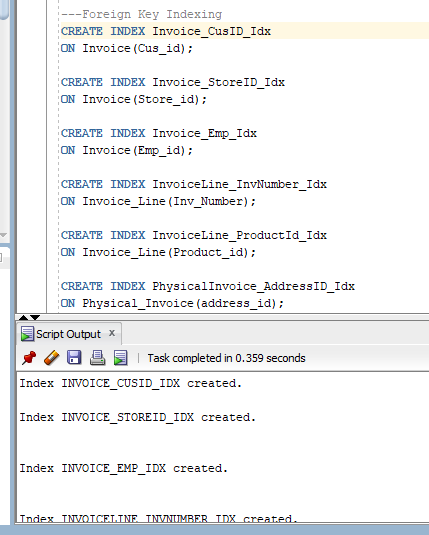
**Foreign Key Index**

|  |
| --- |
| **Employee.StoreCode** |
| **Employee.DeptID** |
| **Administrator.EmpID** |
| **Programmer.EmpID** |
| **Accountant.EmpID** |
| **Invoice.CusID** |
| **Invoice.StoreID** |
| **Invoice.EmpId** |
| **InvoiceLine.InvNumber** |
| **InvoiceLine.ProductID** |
| **OnlineInvoice.InvNumber** |
| **PhysicalInvoice.InvNumber** |
| **PhysicalInvoice.AddressID** |
| **Product.CategoryID** |
| **Product.VenCode** |

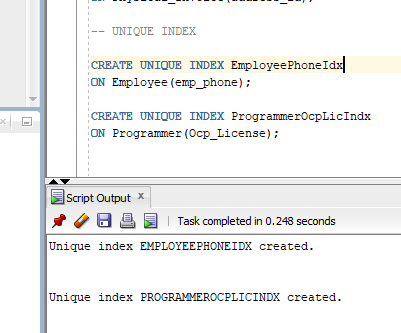
Insofar as foreign keys. We explained below is a table identifying some important foreign keys column, whether or not the index should be unique or not, and why.

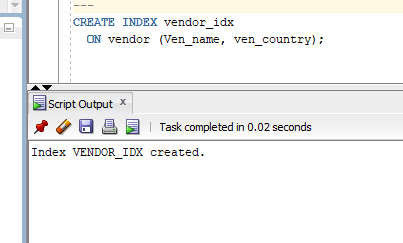
|  |  |  |
| --- | --- | --- |
| **Foreign Key Column** | **Unique/Not Unique** | **Explanation** |
| Invoice.CusID | Not Unique | The foreign key in Invoice referencing Customer is not unique because there can have many Invoice by the same customer |
| Invoice.StoreID | Not Unique | The foreign key in Invoice referencing Store is not unique as there can have many invoices in the same store. |
| Invoice.EmpId | Not Unique | The foreign key in Invoice referencing employee is not unique reason there can have many Invoice by the same employee. |
| InvoiceLine.InvNumber | Not Unique | The foreign key in InvoiceLink referencing Invoice is not unique since a single invoice might be linked to many products. |
| InvoiceLine.ProductID | Not Unique | The foreign key in InvoiceLine referencing Product is not unique because the same product can be purchased many times. |
| PhysicalInvoice.AddressID | Not Unique | The foreign key in PhysicalInvoice referencing Address is not unique because there can be many purchases at the same store location. |

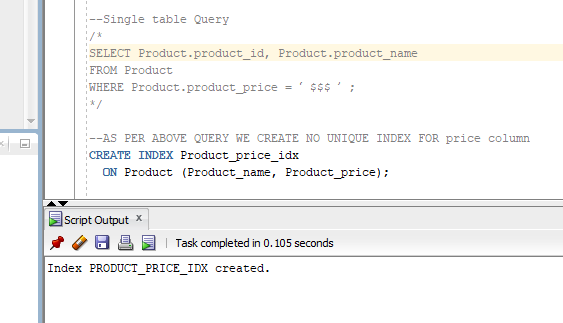
The below screenshot shows some non-unique (columns) foreign keys index creation of my Apparel store DB.



We obtained some unique column, as a result created some unique index as well;

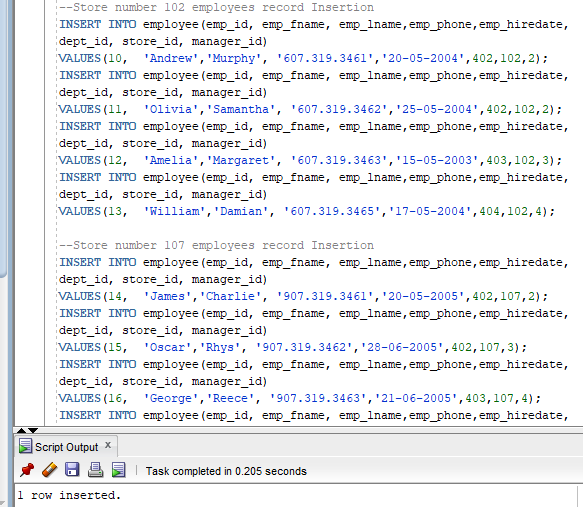


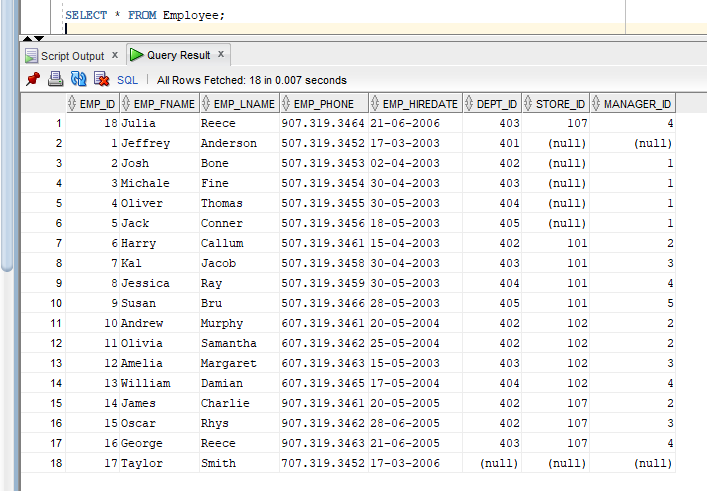
**Query Driven Index**   
  
To retrieve data about supplier we created vendor index as illustrated below;   
  


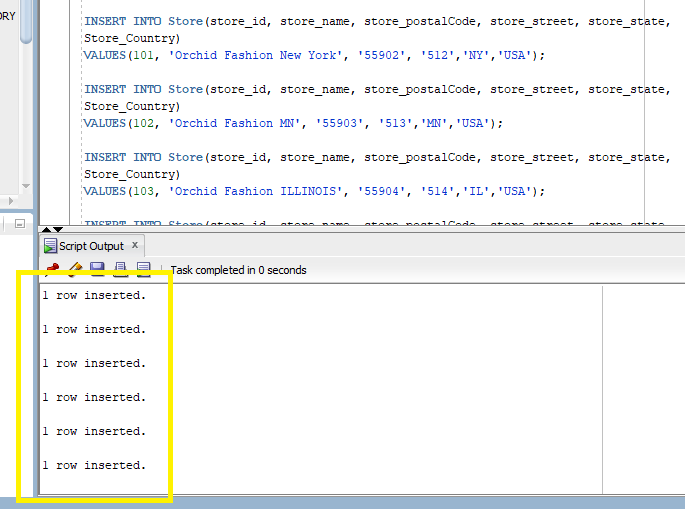
****

**ROW INSERTION**

Please be informed that there are sixteen tables, we would all represent some significant part of data insertion since we are providing entire SQL source code/script, in this way we can save irrelevant explanation.

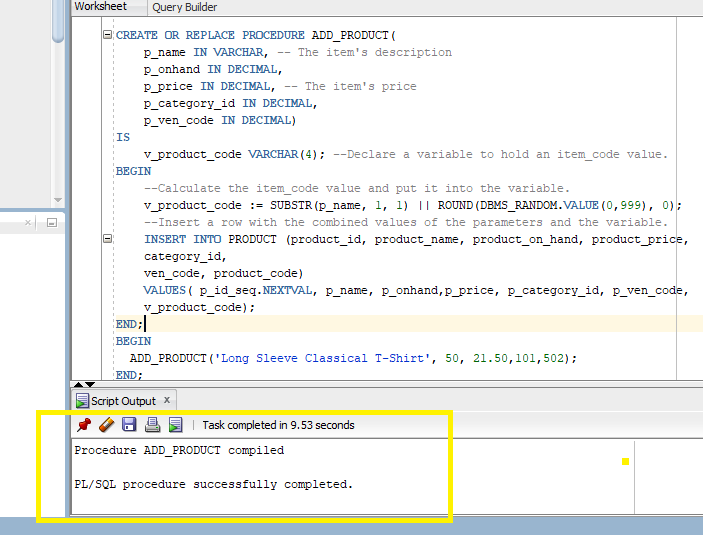
****

****

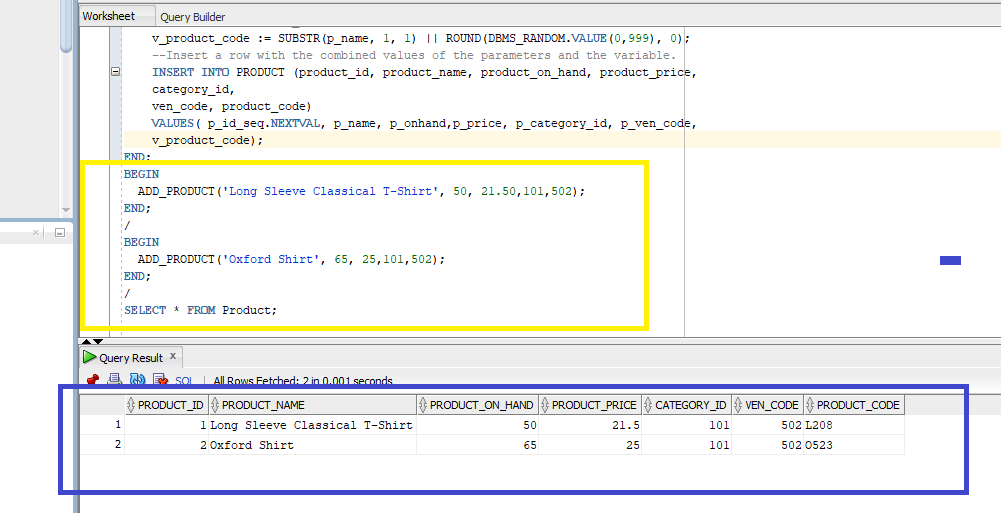
****

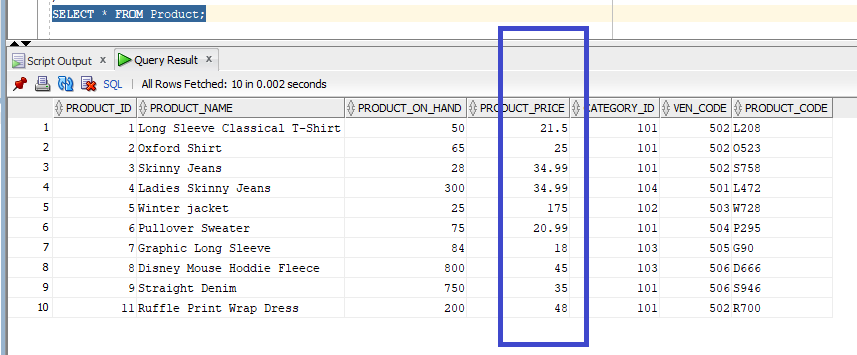
***Reusable, Transaction‐Oriented Store Procedures -1***

ADD product is our one of them use case. Here we create a reusable store procedure named ADD Product. In addition, below screen illustrates that how convenient this store procedure during data insertion. We only need to pass three argument due to functional store procedure rest attributes will generate by developed procedure during invocation.

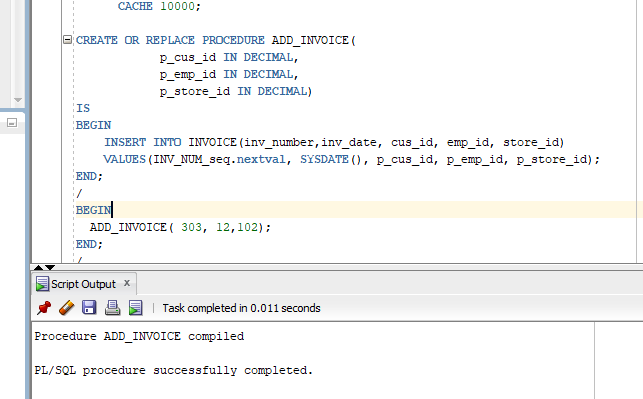
****

Invoking ADD PRODUCE store procedure to add all product

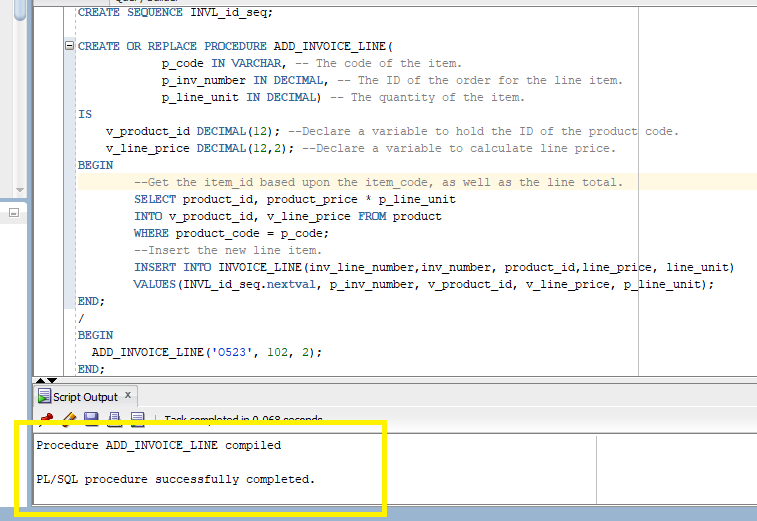
****

****

***Reusable, Transaction‐Oriented Store Procedures -2***

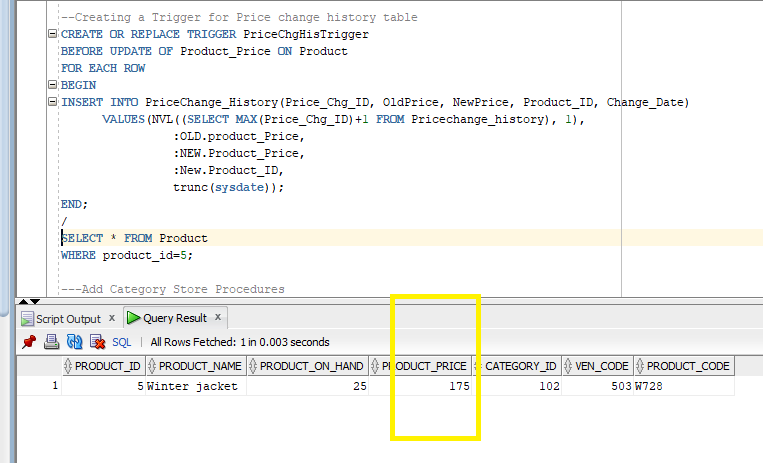
****

***Reusable, Transaction‐Oriented Store Procedures -3***

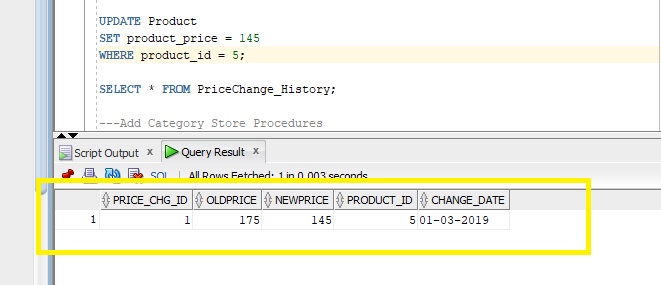
****

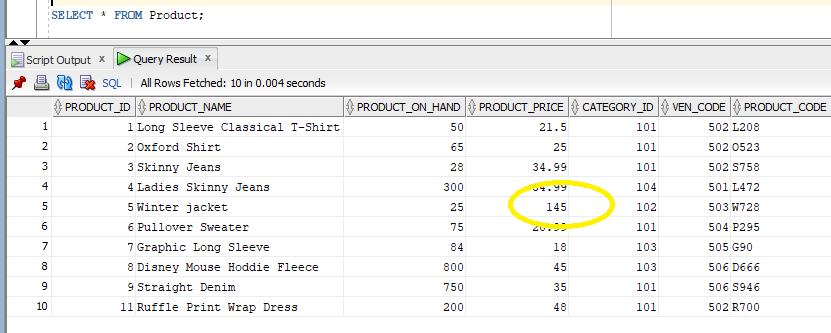
**Creating Trigger**

Price attribute fluctuate frequently so to track the price for specific item we developed a trigger for history table.

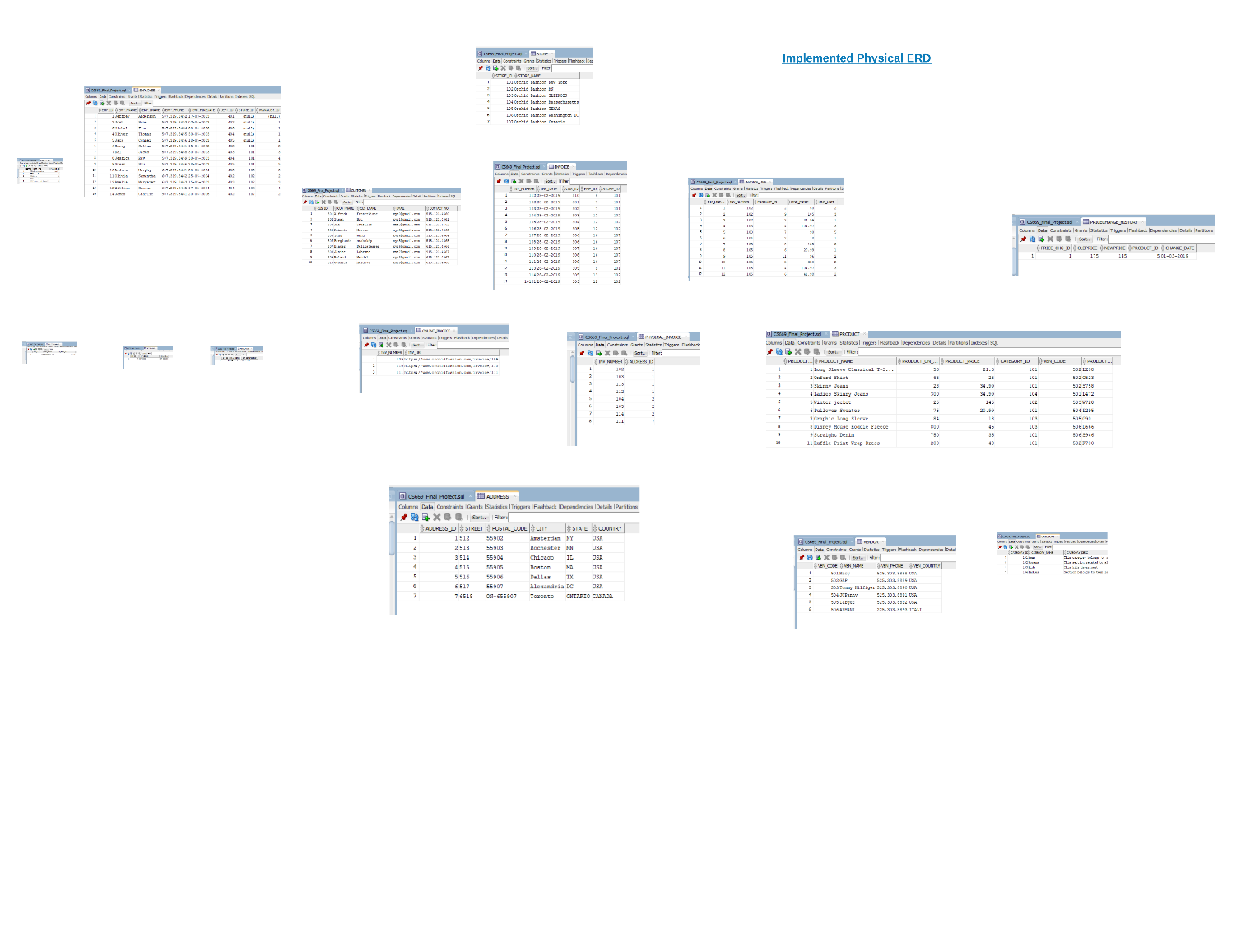


Trigger executed and product table price and price change history both has updated at a time.



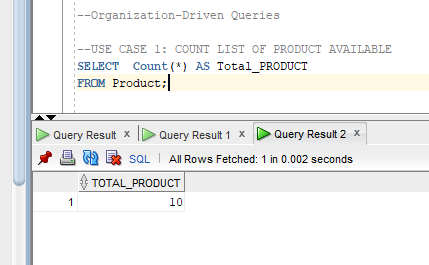


At a glance screenshot of whole database upon rows/data insertion.

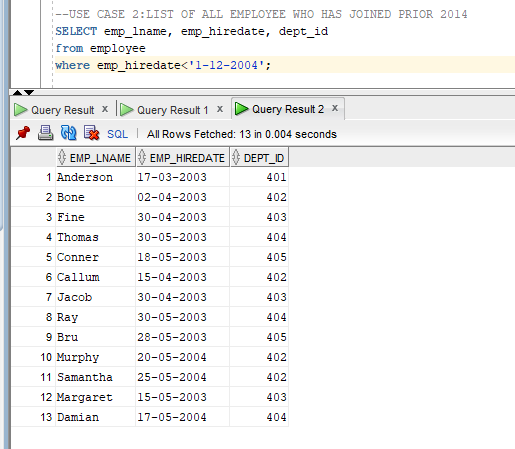


**Organization-Driven Queries**

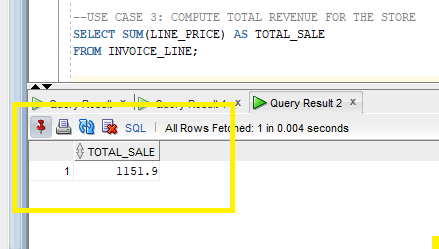
In this stage, we will search some basic quires to evaluate that our database is capable to provide information. These includes aggregate functions, GROUP BY, ORDER BY, and JOIN, Boolean expression, including SUBQURIES.  **USE CASE 1**: COUNT LIST OF PRODUCT AVAILABLE



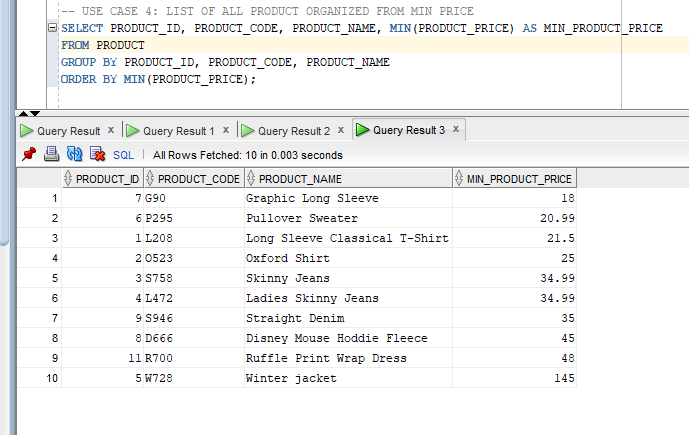
**USE CASE 2**: LIST OF ALL EMPLOYEEs WHO HAVE JOINED PRIOR 2014



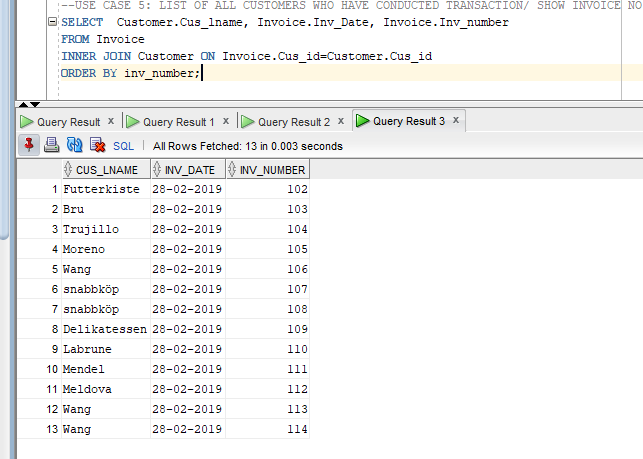
**USE CASE 3**: COMPUTE TOTAL REVENUE



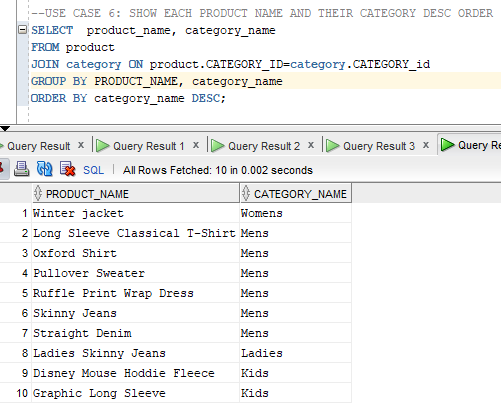
**USE CASE 4**: LIST ALL PRODUCTS ORGANIZED FROM MIN PRICE TO MAX



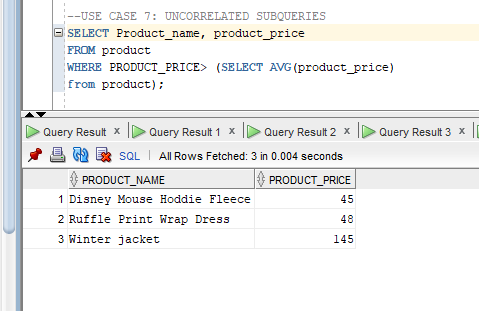
**USE CASE 5**: LIST OF ALL CUSTOMERS WHO HAVE CONDUCTED TRANSACTION/ SHOW INVOICE NO



**USE CASE 6**: SHOW ALL PRODUCT NAME AND THEIR CATEGORY BY DESC ORDER



UNCORRELATED SUBQUERIES: 7 USE CASE



Summary and Reflection

The database I have developed is for an apparel store named ASDMS (Apparel store database management system) which aim is to track and manage the product order and customer.

As final iteration, we gained concreate understanding with regard to own database implementation. We got clarified from beginning stage-design to implementing entire database including high level functionalities, store procedures, relevant trigger, indexing, adding history table to keep the needful information record, subqueries and retrieving information to review that how own database capable.   
  
I am inquiring about one challenges to implement, there is one field named product on hand, it needs automatic updating which mean if a store purchase a product the central product table quantity should be reduce.

Finally, it has been a great learning journey that modeling and development a concreate database is one of the magnificent dexterity of database development.   
  
I greatly appreciated your cooperation and generosity, which made easier for me in this learning journey.

References  
Application link (Under development)   
<https://apex.oracle.com/pls/apex/f?p=101071:LOGIN_DESKTOP:114315914315739>:::::  
  
Attachment   
1. Conceptual ERD adding History Table/entity  
2. Revised Physical ERD

3. SQL source code

Thank You for the review.

Project Iteration 5 End