**Data Model and Database Design**

|  |  |
| --- | --- |
| **Project Name** | Interactive Marketing Agency for Assisting Producers in Selling Seafood Products |
| **Project Supervisor** | Prakash Bhandari  +61 7 3138 1927  [p.bhandari@qut.edu.au](mailto:p.bhandari@qut.edu.au) |
| **Project Manager** | Anirut Yaiyua (n9324356)  +61 426 423 378  [anirut.yaiyua@connect.qut.edu.au](mailto:anirut.yaiyua@connect.qut.edu.au) |
| **Document Date** | April 15th, 2016 |

**\*WHAT HAVE BEEN DONE MUST BE IN PAST TENSES.**

**\*RESULTS MUST BE IN PRESENT TENSES.**

1. **Document Signatures**

|  |  |  |  |
| --- | --- | --- | --- |
| **Role** | **Name** | **Signature** | **Date** |
| Project Manager | Anirut Yaiyua |  |  |
| Project Supervisor | Prakash Bhandari |  |  |

1. **Document Change Control**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version Number** | **Date of Issue** | **Author(s)** | **Brief Description of Change** |
| 1 | 10/04/2016 | Anirut Yaiyua | Creating a ER data model and its explanation |
| 2 | 20/04/2016 | Anirut Yaiyua | Adding symbolic definitions and modifying the SHIPPING\_ADDRESS entity and its relationships |
| 3 | 24/04/2016 | Anirut Yaiyua | Adding Quantity attribute to ORDER and changing the relationship from between CUSTOMER to ORDER to between Lives\_at to ORDER instead |

1. **Definition**

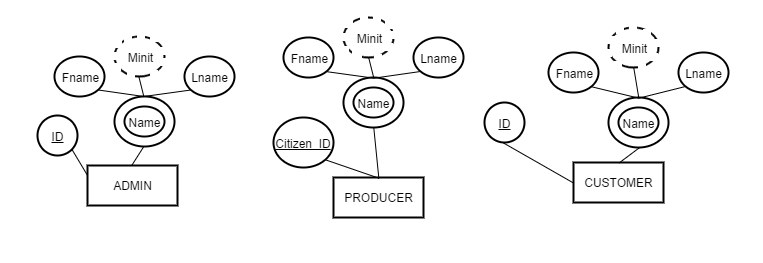
|  |  |
| --- | --- |
| **Term** | **Definition** |
| ER | Entity relationship |
|  |  |

1. **Data Modeling**

To develop a web-based application for producers and customers to trade seafood products online, it is necessary that a database for storing relevant information and business transactions needs to be comprehensively created. Implementing an appropriate database can be completed by initiating a data model which is easily understood by business people and database developers. This is to ensure the correctness of business needs. The most significant step of designing a database is to perform data modeling and it is also vital that the created data model is subsequently verified to determine whether it has been correctly developed based on the requirement specifications (Umanath & Scamell, 2007). In this project, the business scopes and requirements previously developed will be analyzed to create an Entity-Relationship data model. Once the ER data model is fully developed, the actual database can be systematically implemented. The ER data model necessarily created for database implementation including symbolic definitions, according to Umanath & Scamell (2007), are illustrated below.

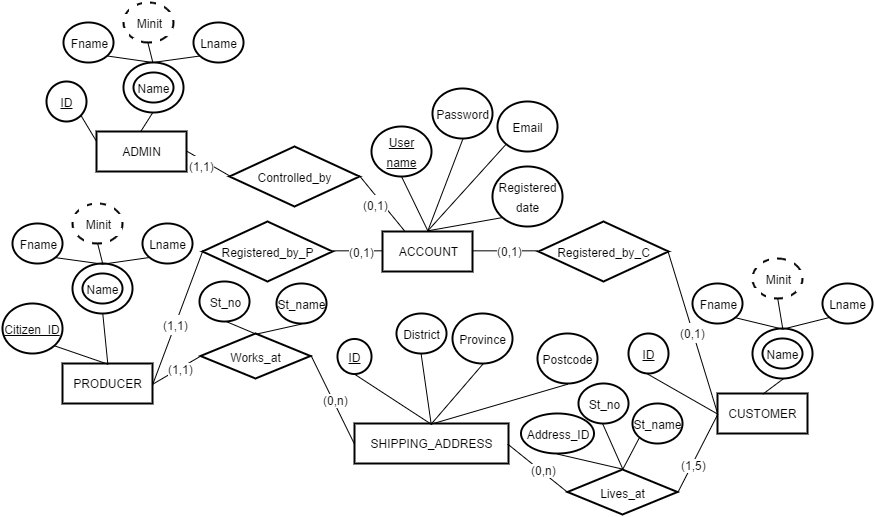
|  |  |
| --- | --- |
| **Symbols** | **Definitions** |
| C:\Users\thinkpop\Desktop\QUT\IFN701 and IFN702\Documents\Data Model\Model Pictures\Symbol-Entity type.png | Entity type |
| C:\Users\thinkpop\Desktop\QUT\IFN701 and IFN702\Documents\Data Model\Model Pictures\Symbol-Weak entity type.png | Weak entity type |
| C:\Users\thinkpop\Desktop\QUT\IFN701 and IFN702\Documents\Data Model\Model Pictures\Symbol-Relationship type.png | Relationship type |
| C:\Users\thinkpop\Desktop\QUT\IFN701 and IFN702\Documents\Data Model\Model Pictures\Symbol-Identifying relationship type.png | Identifying relationship type |
| C:\Users\thinkpop\Desktop\QUT\IFN701 and IFN702\Documents\Data Model\Model Pictures\Symbol-Attribute with optional value.png | Attribute with optional value |
| C:\Users\thinkpop\Desktop\QUT\IFN701 and IFN702\Documents\Data Model\Model Pictures\Symbol-Attribute with mandatory value.png | Attribute with mandatory value |
| C:\Users\thinkpop\Desktop\QUT\IFN701 and IFN702\Documents\Data Model\Model Pictures\Symbol-Unique identifier.png | Unique identifier |
| C:\Users\thinkpop\Desktop\QUT\IFN701 and IFN702\Documents\Data Model\Model Pictures\Symbol-Multi-valued attribute.png | Multi-valued attributes |

*Table 4.1 – Demonstrates symbols and definitions used to illustrate the ER data model*



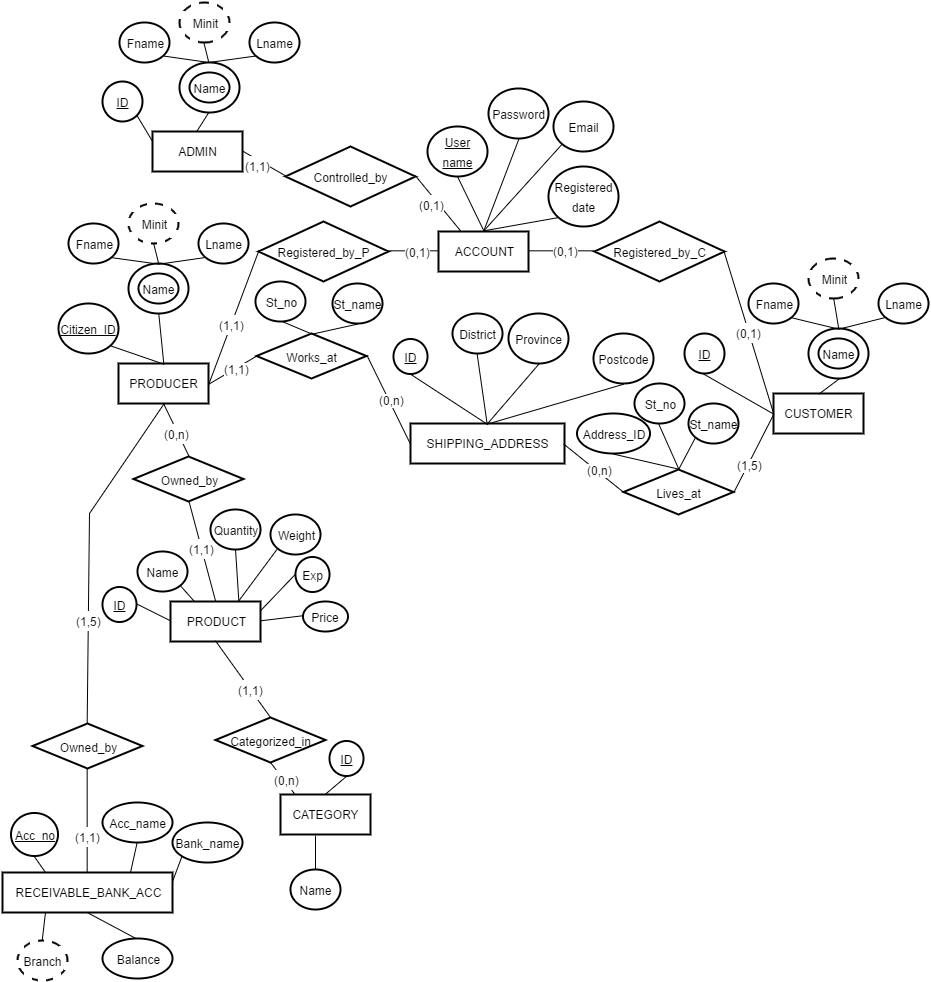
*Figure 4.1 – Illustrates three entities including ADMIN, PRODUCER and CUSTOMER with their attributes*

The cooperative organization consists of three main groups of stakeholders using the solution including producers, customers and administrators. These entities contain similar attributes which are ID, Name as a multi-valued attribute, First name, Middle name as an optional attribute and Last name. However, the PRODUCER entity has a different ID called Citizen ID which is the real identification number of a producer while the others uses the ID generated by the system.



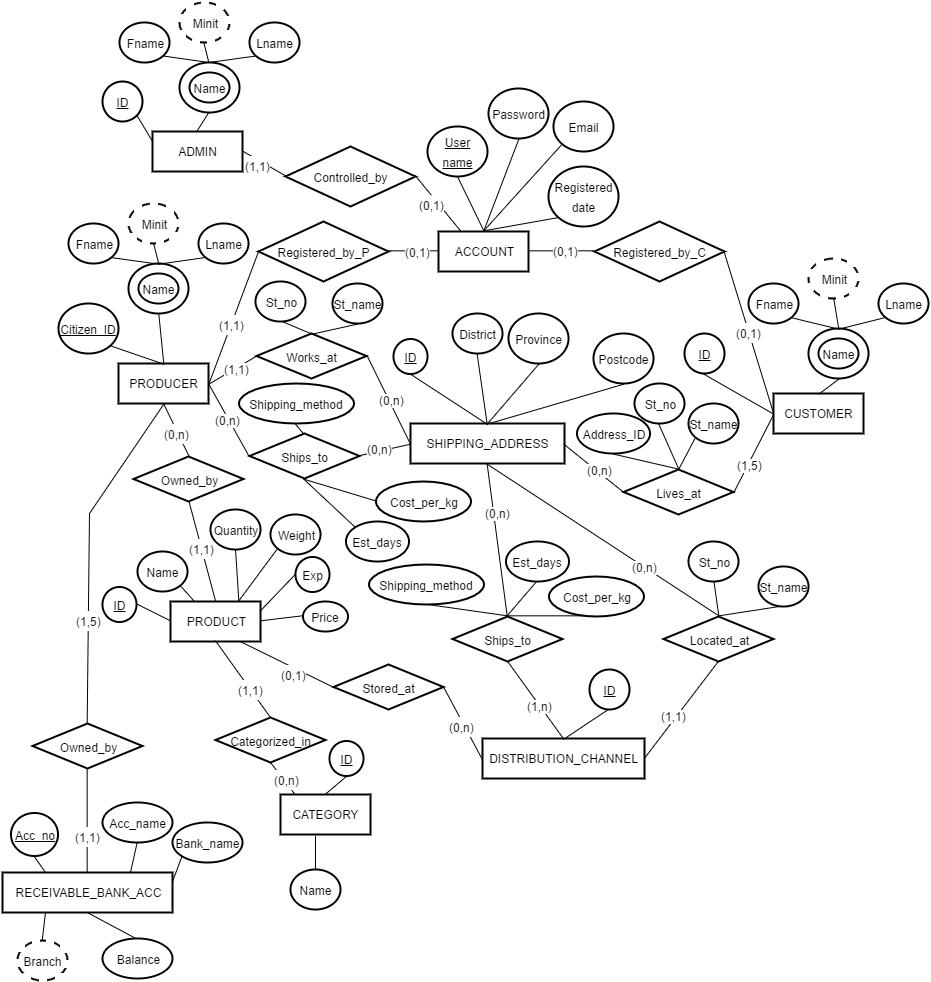
*Figure 4.2 – Shows two additional entities including ACCOUNT and SHIPPING\_ADDRESS and relationships between them*

The above picture illustrates the new two entities called ACCOUNT and SHIPPING\_ADDRESS. The ACCOUNT entity is created to store information about account details that stakeholders use to access the system. This entity contains unique username, password, email and registered date attributes. The relationships between PRODUCER and ACCOUNT and between ADMIN and ACCOUNT are similar in which a producer or an admin must have at least and only one account. However, the relationship between CUSTOMER and ACCOUNT is a little different as a customer can have only one account but it is acceptable for a customer not to have any account. This answers one of business requirements stating that customers can purchase products without having to register for an account. Another new entity is SHIPPING\_ADDRESS. This entity is responsible for storing information regarding specific areas in Thailand which includes shipping address ID, District, Province and Postcode. The SHIPPING\_ADDRESS entity in the above picture is linked to the PRODUCER and CUSTOMER entities in order to describe where producers or customers live. Moreover, it is noticed that a producer must work at one shipping address while a customer can have at least one and up to five addresses. The relationship between the PRODUCER and SHIPPING\_ADDRESS results in an occurrence of two new attributes including Street number and Street name. Similarly, the relationship between SHIPPING\_ADDRESS and CUSTOMER also leads to an emergence of those two attributes plus an address ID to ensure that a customer can have more than one address. This SHIPPING\_ADDRESS entity will be lately linked to other entities for delivery process demonstration as it is a main entity containing all of the particular location identified by unique sets of a district, province and postcode.



*Figure 4.3 – Displays three new entities involving PRODUCT, RECEIVABLE\_BANK\_ACC and CATEGORY and their relationships*

The new entities added into the ER data model are PRODUCT, CATEGORY and RECEIVABLE\_BANK\_ACC. The PRODUCT entity is linked to PRODUCER because a producer can determine a list of products to. In other words, a producer can have more than one products while each product must belong to only one producer. The attributes owned by the PRODUCT entity are product ID, Name, Quantity, Weight, Expiry Date and Price. Moreover, a product must be assigned with only one category. So it can be seen that PRODUCT is related to CATEGORY which contains category ID and Name attributes. Another entity is RECEIVABLE\_BANK\_ACC. This entity is used to store bank account details of producers for receiving payments from customers who wish to purchase their products. The attributes of this entity include Account Number, Account Name, Bank Name, Branch and Balance. The relationship between PRODUCER and RECEIVABLE\_BANK\_ACC indicates that a producer can have up to five bank accounts while each bank account must be owned by only one producer.



*Figure 4.4 – Shows a new entity called DISTRIBUTION\_CHANNEL and its relationships plus relationships between SHIPPING\_ADDRESS and other entities*

The new entity which is DISTRIBUTION\_CHANNEL has been added to the ER data model. This entity contains only an ID attribute. This entity is linked to the PRODUCT entity to illustrate that a product may be kept at a distribution channel and a distribution channel can store many products. Another relationship of DISTRIBUTION\_CHANNEL is with SHIPPING\_ADDRESS. This means a distribution channel must be assigned with only one shipping address to describe where the distribution channel is located. This relationship leads to a happening of two new attributes including Street Number and Street Name. In addition, a new relationship between PRODUCER and SHIPPING\_ADDRESS called Ships\_to has been added in order to depict that a producer can determine the specific areas that he or she can manage to deliver products based on the locations listed in SHIPPING\_ADDRESS. Similarly, the relationship between SHIPPING\_ADDRESS and DISTRIBUTION\_CHANNEL, the Ships\_to relationship between them shows that a distribution channel can be assigned with many locations where it can distribute stored products to. These new relationships create three attributes involving Shipping Method, Estimated Days and Shipping Cost.

C:\Users\thinkpop\Desktop\QUT\IFN701 and IFN702\Documents\Data Model\Model Pictures\Interactive Marketing Agency.png

*Figure 4.5 – Illustrates the ORDER entity and its relationships*

The last entity called ORDER with three attributes including Order Number, Order Date, and Quantity has been added to the ER data model. This entity is linked to PRODUCT to describe what products are contained in each order. Therefore, it means that an order can be combined with many products while a product can be stored in only one order. Also, ORDER is related to the CUSTOMER entity. This implies that each order must be owned by only one customer.

Moreover, it is noticed that there are more important details that cannot be included in the ER data model illustrated above. Therefore, the sematic integrity constraints for the Coarse-granular Design-Specific ER model as well as Entity-Level Business Rules must be created according to Umanath & Scamell (2007).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Entity/Relationship**  **Type Name** | **Attribute Name** | **Data Type** | **Size** | **Domain Constraints** |
| ADMIN | ID | Numeric | - |  |
| ADMIN | Fname | Alphabetic | 30 |  |
| ADMIN | Minit | Alphabetic | 30 |  |
| ADMIN | Lname | Alphabetic | 30 |  |
| PRODUCER | Citizen\_ID | Alphanumeric | 13 | 13-digit numeric values |
| PRODUCER | Fname | Alphabetic | 30 |  |
| PRODUCER | Minit | Alphabetic | 30 |  |
| PRODUCER | Lname | Alphabetic | 30 |  |
| CUSTOMER | ID | Numeric | - |  |
| CUSTOMER | Fname | Alphabetic | 30 |  |
| CUSTOMER | Minit | Alphabetic | 30 |  |
| CUSTOMER | Lname | Alphabetic | 30 |  |
| ACCOUNT | User\_name | Alphanumeric | 12 |  |
| ACCOUNT | Password | Alphanumeric | 12 |  |
| ACCOUNT | Email | Alphanumeric | 80 | Must contain a sign @ sign |
| ACCOUNT | Registered\_date | Date | 8 |  |
| SHIPPING\_ADDRESS | ID | Numeric | - |  |
| SHIPPING\_ADDRESS | District | Alphabetic | 30 |  |
| SHIPPING\_ADDRESS | Province | Alphabetic | 30 |  |
| SHIPPING\_ADDRESS | Postcode | Numeric | 5 |  |
| Works\_at | St\_no | Alphanumeric | 30 |  |
| Works\_at | St\_name | Alphanumeric | 30 |  |
| Lives\_at | Address\_ID | Numeric | - |  |
| Lives\_at | St\_no | Alphanumeric | 30 |  |
| Lives\_at | St\_name | Alphanumeric | 30 |  |
| RECEIVABLE\_BANK\_ACC | Acc\_no | Alphanumeric | 10 | 10-digit numeric values |
| RECEIVABLE\_BANK\_ACC | Acc\_name | Alphabetic | 30 |  |
| RECEIVABLE\_BANK\_ACC | Bank\_name | Alphabetic | 30 |  |
| RECEIVABLE\_BANK\_ACC | Branch | Alphabetic | 30 |  |
| RECEIVABLE\_BANK\_ACC | Balance | Numeric | (8.2)\* |  |
| PRODUCT | ID | Numeric | - |  |
| PRODUCT | Name | Alphabetic | 30 |  |
| PRODUCT | Quantity | Numeric | - |  |
| PRODUCT | Weight | Numeric | (3.2)\* | Decimal values from 0.01 to 3.00 |
| PRODUCT | Expiry\_date | Date | 8 |  |
| PRODUCT | Price | Numeric | (8.2)\* |  |
| CATEGORY | ID | Numeric | - |  |
| CATEGORY | Name | Alphabetic | 30 |  |
| DISTRIBUTION\_CHANNEL | ID | Numeric | - |  |
| Located\_at | St\_no | Alphanumeric | 30 |  |
| Located\_at | St\_name | Alphanumeric | 30 |  |
| Ships\_to (PRODUCER) | Shipping\_method | Alphabetic | 30 |  |
| Ships\_to (PRODUCER) | Est\_days | Numeric | - |  |
| Ships\_to (PRODUCER) | Cost\_per\_kg | Numeric | (8.2)\* |  |
| Ships\_to (DISTRIBUTION\_CHANNEL) | Shipping\_method | Alphabetic | 30 |  |
| Ships\_to (DISTRIBUTION\_CHANNEL) | Est\_days | Numeric | 4 |  |
| Ships\_to (DISTRIBUTION\_CHANNEL) | Cost\_per\_kg | Numeric | (8.2)\* |  |
| ORDER | Order\_no | Numeric | - |  |
| ORDER | Quantity | Numeric | - |  |
| ORDER | Order\_date | Date | 8 |  |

*Table 4.2 – Demonstrates constraints of attributes within the data model*

**Entity-Level Business Rules**

1. An administrator must have only one user account.
2. A producer is identified by his or her citizen ID and each of them must have only one user account.
3. A producer must have at least one and no more than five bank accounts for receiving payments from customers while a bank account must be owned by only one producer.
4. A producer must have one address in order to state where he or she works.
5. A producer can own many products while a product can be owned by only one producer.
6. A product must be assigned to only one category and can be assigned to one distribution channel if a producer who owns the product chooses to use the inventory and delivery service.
7. A producer or a distribution channel can contain many shipping addresses to describe where they can deliver products to.
8. A product must not have weight over than 3 kilograms.
9. A customer can have none or one user account.
10. A customer can have at least one but no more than five addresses to obtain products.
11. An order must contain at least one product but can be owned by only one customer.
12. To sell a product, the producer must have at least one bank account and one product with at least one specified shippable address.
13. To purchase a product, the customer must have identified his or her details and at least one receiving address.
14. **Database Implementation**

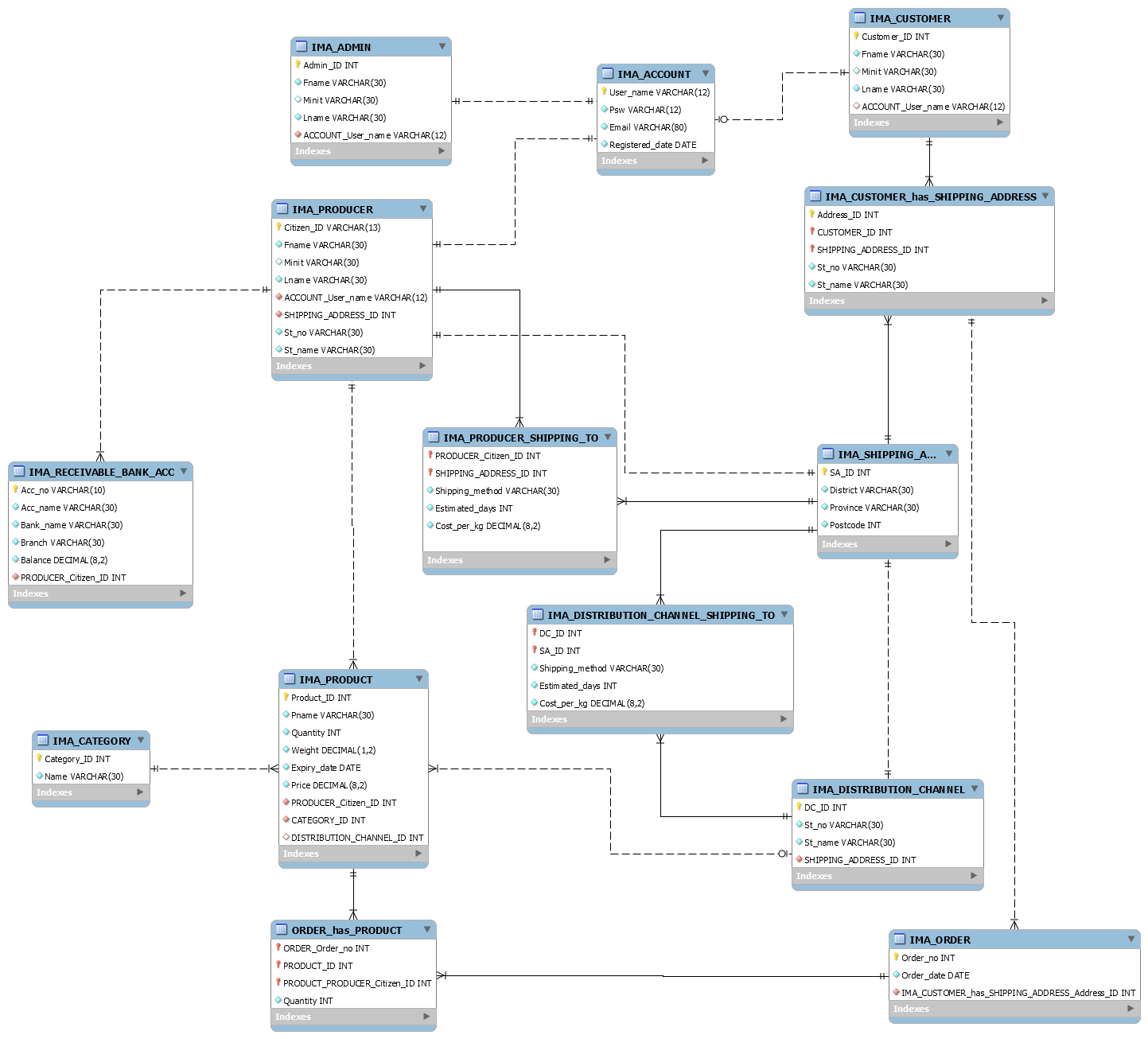
After the draft version of ER model, the list of constraints and the set of business rules have been developed, the actual ER diagram can be drawn by using MySQL Workbench. This tool helps database designers in creating an Entity-Relationship diagram in a more effective way. By using this tool, database developers are not required to perform a manual action to create entity relationships because it provides a user-friendly interface that they can easily create a relationship by drawing a line between entities. After the relationship of entities has been drawn, the tool will automatically generate foreign keys or even a new table depending on the type of relationship created. Once the ER diagram has been fully developed, database developers can simply generate a SQL script from the ER diagram by using an export feature provided in the tool. This script will be used to create tables in the database. These are the reasons why MySQL Workbench is chosen to facilitate this project. However, there is a few limitations of MySQL Workbench as well. One of them is that the necessary constraints cannot be defined within the ER diagram drawn in the tool. This issue will be mitigated by manually adding constraints into the SQL script file.

**5.1 Database Creation**

It can be seen in the ER diagram (figure 5.1) that all of the relationship types represented in diamond shapes of the draft version of ER model as shown in figure 4.5 have all been converted to tables and look similar to the entities types. This is because some relationship types contain their own attributes, so it is necessary to consider them as tables too and MySQL Workbench handles this convert for us. Moreover, the definitions of symbols and relationships of the ER diagram created by using MySQL Workbench are described in the table below.

|  |  |
| --- | --- |
| **Symbols** | **Definitions** |
| C:\Users\thinkpop\Desktop\QUT\IFN701 and IFN702\Documents\Data Model\Model Pictures\MySQL Workbench - one-to-one-relationship-identifying.png | One-to-one identifying relationship |
| C:\Users\thinkpop\Desktop\QUT\IFN701 and IFN702\Documents\Data Model\Model Pictures\MySQL Workbench - one-to-one-relationship-non-identifying.png | One-to-one non-identifying relationship |
| C:\Users\thinkpop\Desktop\QUT\IFN701 and IFN702\Documents\Data Model\Model Pictures\MySQL Workbench - one-to-many-relationship-identifying.png | One-to-many identifying relationship |
| C:\Users\thinkpop\Desktop\QUT\IFN701 and IFN702\Documents\Data Model\Model Pictures\MySQL Workbench - one-to-many-relationship-non-identifying.png | One-to-many non-identifying relationship |
| C:\Users\thinkpop\Desktop\QUT\IFN701 and IFN702\Documents\Data Model\Model Pictures\MySQL Workbench - many-to-many-relationship-identifying.png | Many-to-many identifying relationship |
| C:\Users\thinkpop\Desktop\QUT\IFN701 and IFN702\Documents\Data Model\Model Pictures\MySQL Workbench  - primary-key.png | Primary key |
| C:\Users\thinkpop\Desktop\QUT\IFN701 and IFN702\Documents\Data Model\Model Pictures\MySQL Workbench  - primary-referrence-key.png | Primary key referenced to a value of a primary key in another table |
| C:\Users\thinkpop\Desktop\QUT\IFN701 and IFN702\Documents\Data Model\Model Pictures\MySQL Workbench  - attribute-mandatory.png | Attribute with mandatory value |
| C:\Users\thinkpop\Desktop\QUT\IFN701 and IFN702\Documents\Data Model\Model Pictures\MySQL Workbench  - attribute-optional.png | Attribute with optional value |
| C:\Users\thinkpop\Desktop\QUT\IFN701 and IFN702\Documents\Data Model\Model Pictures\MySQL Workbench  - referrence-attribute-mandatory.png | Attribute with mandatory value referenced to a value of a primary key from another table |
| C:\Users\thinkpop\Desktop\QUT\IFN701 and IFN702\Documents\Data Model\Model Pictures\MySQL Workbench  - referrence-attribute-optional.png | Attribute with optional value referenced to a value of a primary key from another table |

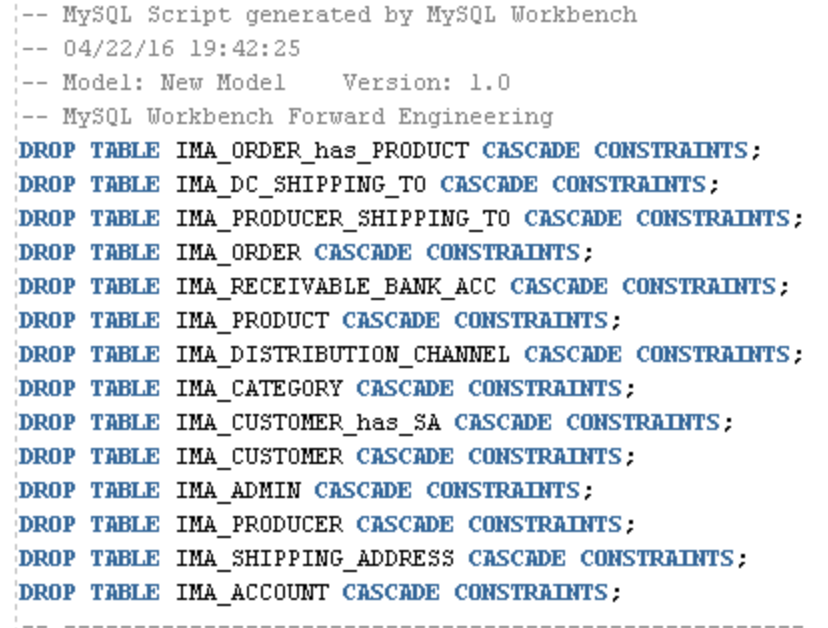
*Table 5.1 – Explains definitions of entity and relationship symbols used to create an ER diagram on MySQL Workbench*



*Figure 5.1 – Illustrates the ER diagram created by using MySQL Workbench*

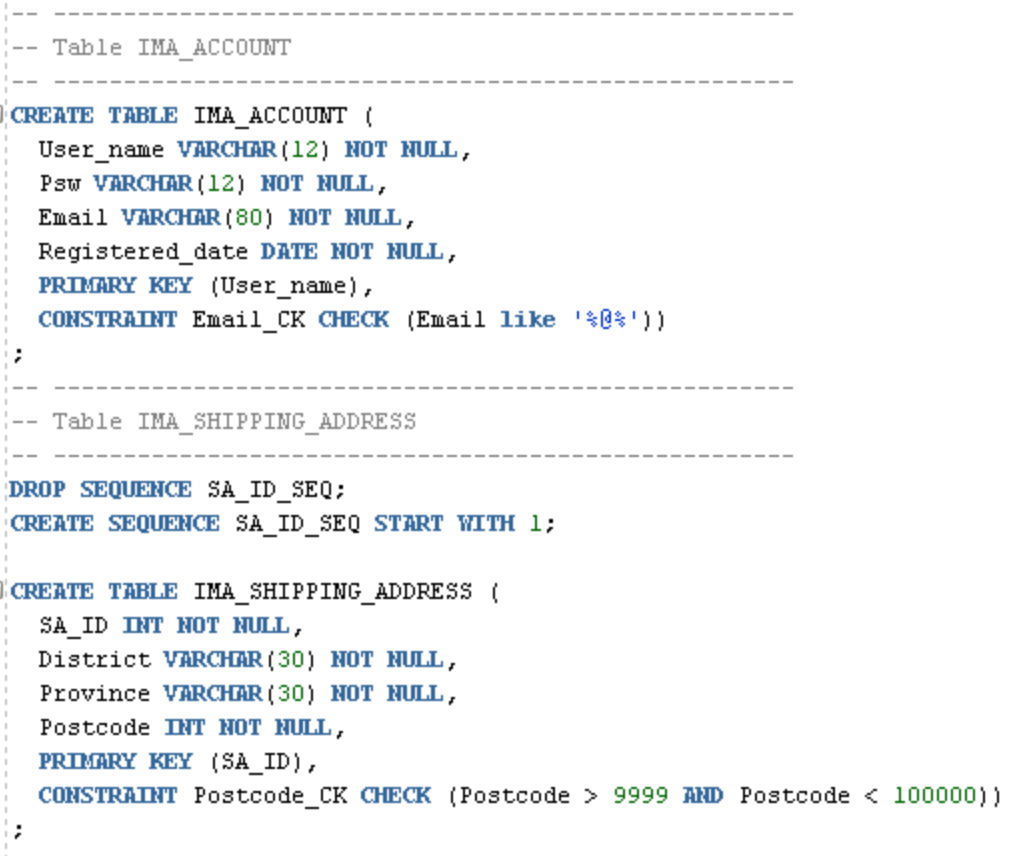
It can be seen in figure 5.1 that the relationship types consisting of attributes are converted into tables. Similarly, the many-to-many relationship types in the draft ER model are turned into tables as well, regardless of having their own attributes. In this diagram, all relationships are identified using foreign keys. A foreign key is used to contain a value that referenced to a value of a primary key in another table according to Powell (2007). For example, the IMA\_PRODUCT table contains CATEGORY\_ID attribute as a foreign key that references to the primary key called Category\_ID in the IMA\_CATEGORY table. This means each product must be assigned with one category as described in the requirement specification. Moreover, in the ER diagram shown in figure 5.1, some of constraints specified in table 4.2 are determined in tables. The Fname attribute in the table named IMA\_ADMIN, for instance, must not contain more than 30 characters. This constrain is specified in the diagram using statement “Fname VARCHAR(30)”. However, there some constraints that cannot be determined in the diagram. For example, an email must contain an at sign. These constraints will be manually added into the SQL script generated from this ER diagram. MySQL Workbench facilitates database developers by providing a feature that they can use to automatically generate a SQL script file used to create tables. This can be done by going to “File>Export>Forward Engineer SQL CREATE Script…” or pressing “Ctrl+Shift+G” in MySQL Workbench. More optional details for creating the SQL script will be asked.

Firstly, the tool generates DROP TABLE statement to ensure that every time a database developer executes the script file, all existing tables will be deleted before being recreated. These statements are shown in the figure 5.2.



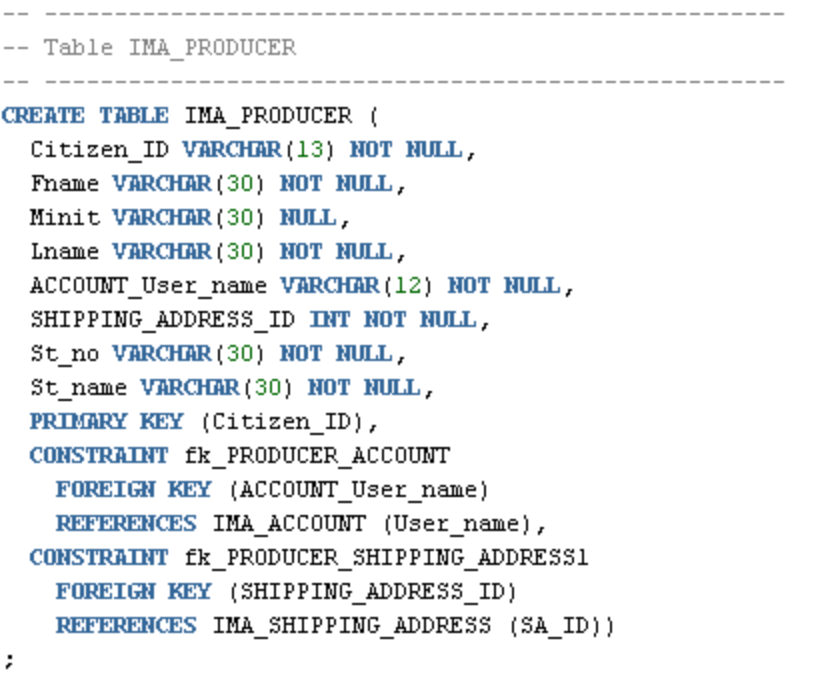
*Figure 5.2 – Illustrates DROP TABLE statements in the SQL CREATE script*

According to figure 5.2, it is noticed that the sequence of tables that are dropped is orderly opposite to the set of CREATE TABLE statements. This is to ensure that the table without a primary key referencing to another table will be deleted first because the system prevents the loss of information. For example, if a product A contains category id valued 1 in the IMA\_PRODUCT table and the IMA\_CATEGORY table can be dropped before deleting IMA\_PRODUCT, this will lead to the loss of information as it is not possible to find exactly what category the product A is assigned with because all categories have already been deleted. Moreover, the CASCADE CONSTRAINTS parameters are added to the DROP TABLE statements in order to remove constraints in the tables, otherwise the system will show an error message if the table with constraints is dropped without this parameter (Powell, 2007).

****

*Figure 5.3 – Shows the CREATE TABLE statements of the IMA\_ACCOUNT and IMA\_SHIPPING\_ADDRESS tables respectively*

It can be seen in figure 5.3 that the Email\_CK constraint is added to the IMA\_ACCOUNT table in order to ensure the correct format of email addresses according to the business specification. Similarly, the constraint named Postcode\_CK is included in the CREATE TABLE statement of table IMA\_SHIPPPING\_ADDRESS to determine a fixed number of digits for Postcode values. The figure 5.3 also illustrates that User\_name and SA\_ID attributes are the primary keys of the IMA\_ACCOUNT and IMA\_SHIPPING\_ADDRESS tables respectively.

**

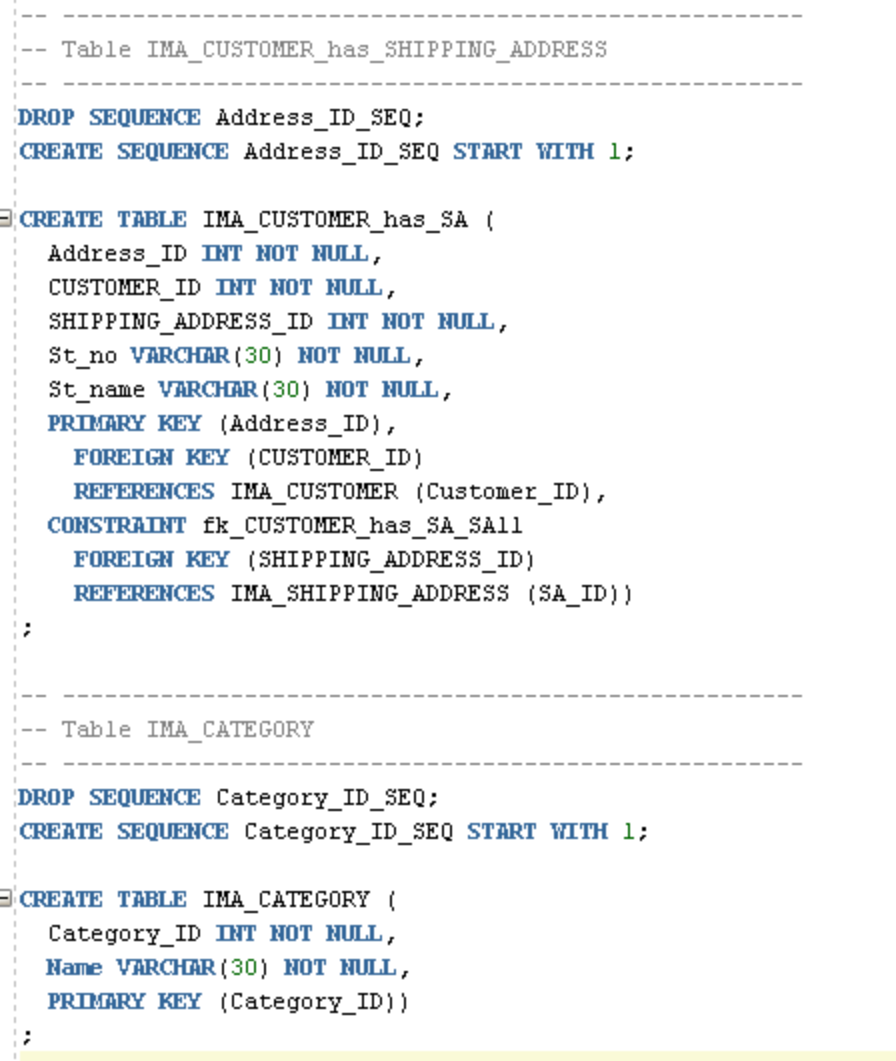
*Figure 5.4 – Shows the CREATE TABLE statements of the IMA\_PRODUCER table*

In figure 5.4, the primary key of the IMA\_PRODUCER table is Citizen\_ID while there are two foreign keys that reference to other tables. The first is ACCOUNT\_User\_name. This attribute references to a primary key of the IMA\_ACCOUNT table. Another is the SHIPPING\_ADDRESS\_ID which references to the primary key contained in the IMA\_SHIPPING\_ADDRESS table.

****

*Figure 5.5 – Shows the CREATE TABLE statements of the IMA\_ADMIN and IMA\_CUSTOMER tables respectively*

These two tables are similarly structured where they have ID as their primary keys and ACCOUNT\_User\_name as their foreign keys. The difference of these two tables is that the ACCOUNT\_User\_name attribute accepts blank values as stated in the business specification that a customer can purchase products without having to registering a user account.



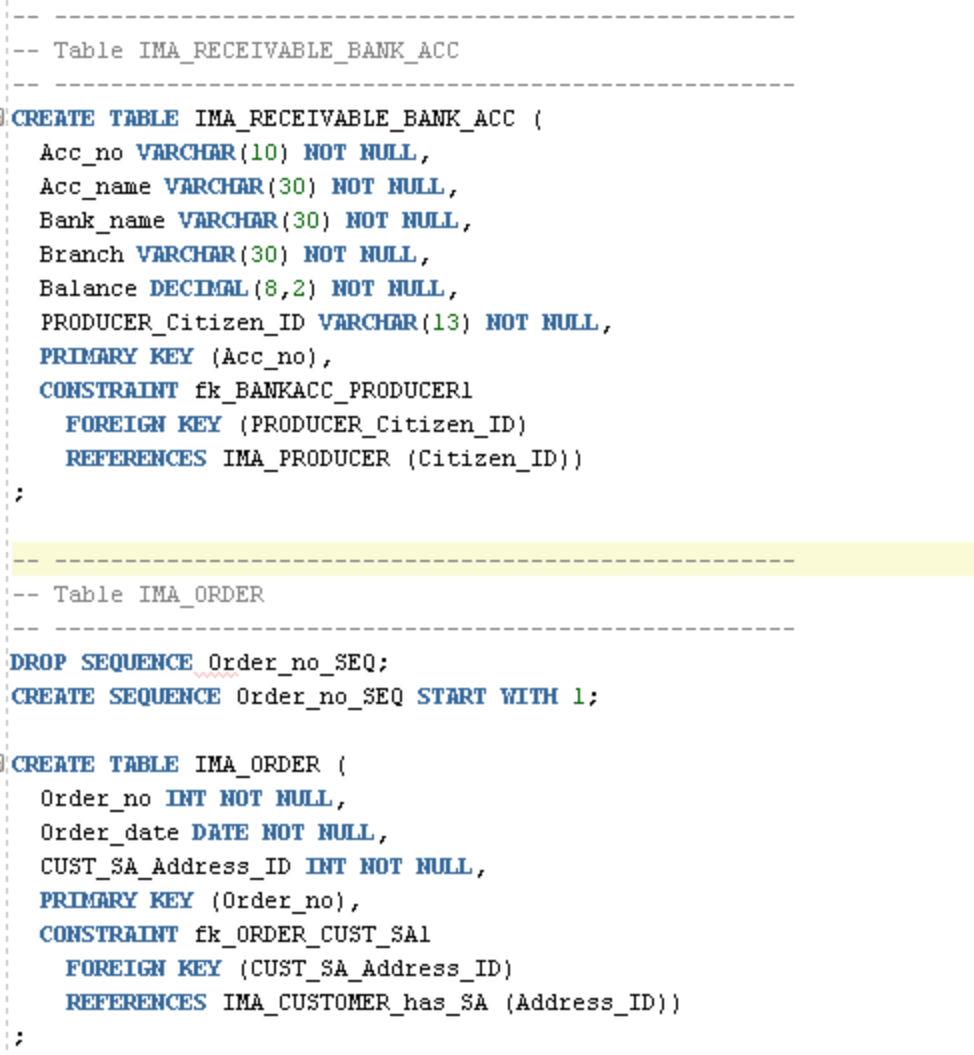
*Figure 5.6 – Shows the CREATE TABLE statements of the IMA\_CUSTOMER\_has\_SA and IMA\_CATEGORY tables*

The IMA\_CUSOMTER\_has\_SA table represents the relationship between IMA\_CUSTOMER and IMA\_SHIPPING\_ADDRESS tables. This table contains the Address\_ID attribute as a primary key and each record is assigned with a customer ID as a foreign key. However, IMA\_CATEGORY is the table that does not contain foreign keys as its data is not dependent on data in the other tables.



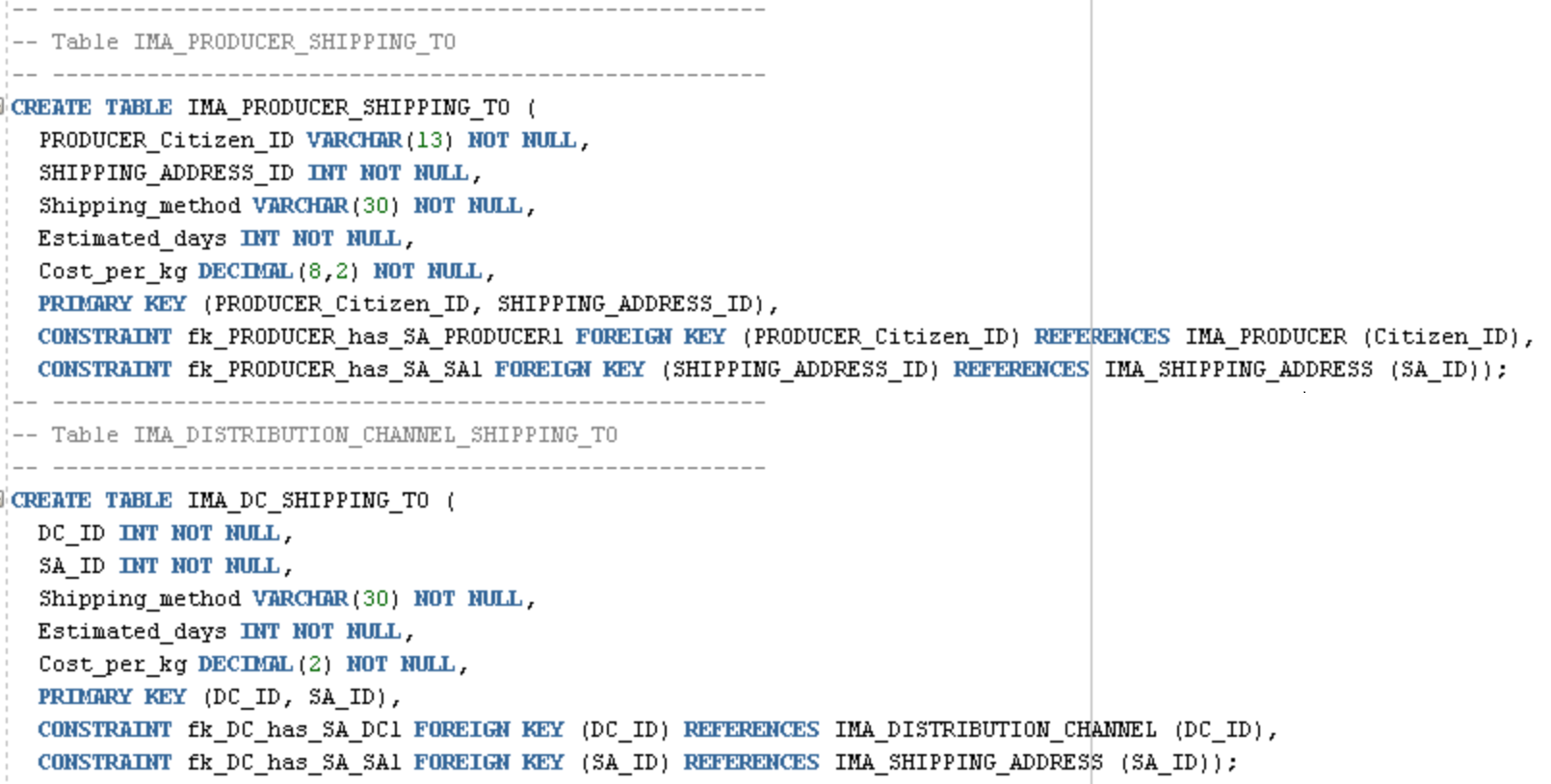
*Figure 5.7 – Shows the CREATE TABLE statements of the IMA\_DISTRIBUTION\_CHANNEL and IMA\_PRODUCT tables*

According to figure 5.7, the IMA\_DISTRIBUTION\_CHANNEL table is created with DC\_ID as a primary key and SHIPPING\_ADDRESS\_ID as a foreign key referencing to the IMA\_SHIPPING\_ADDRESS. The IMA\_PRODUCT table has Product\_ID as a primary key and contains three foreign keys including PRODUCER\_Citizen\_ID, CATEGORY\_ID, and DISTRIBUTION\_CHANNEL\_ID. The PRODUCER\_Citizen\_ID is used to identify the owner of the product. The CATEGORY\_ID is for specifying what category the product is in and the DISTRIBUTION\_CHANNEL\_ID is to determine which distribution channel the product is stored within. However, the DISTRIBITION\_CHANNEL is the only attribute that can contain null value because the business specification mentions that a producer who owns a product can choose whether to store the product with the organization or him/herself.

**

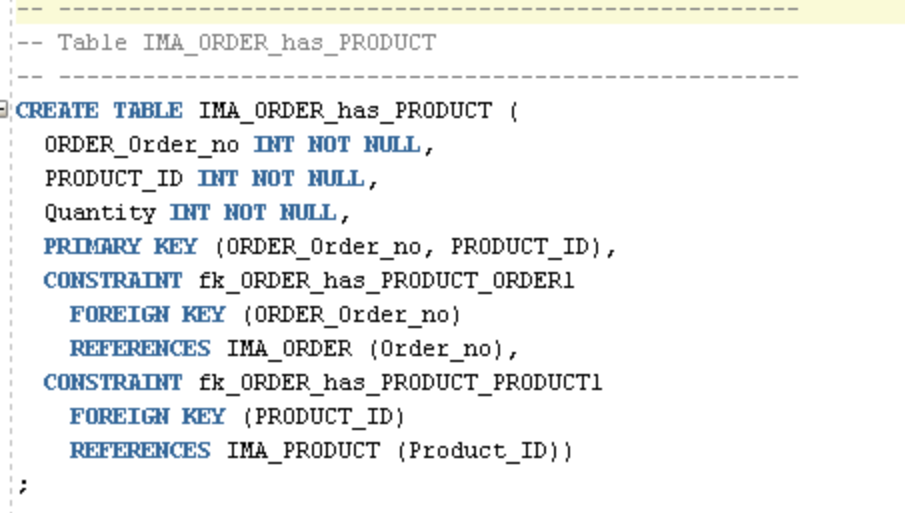
*Figure 5.8 – Shows the CREATE TABLE statements of the IMA\_RECEIVABLE\_BANK\_ACC and IMA\_ORDER tables*

The IMA\_RECEIVABLE\_BANK\_ACC table is created with a single primary key named Acc\_no. This unique identifier must contain 10 digits as specified in the business requirements. There is one foreign key of this table which is PRODUCER\_Citizen\_ID to identify whom the bank account belongs to. The IMA\_ORDER table is established with Order\_no as its primary key while the foreign key of this table is CUST\_SA\_Address\_ID to determine where this order will be delivered to. This attribute references to Address\_ID contained in the IMA\_CUSOMER\_has\_SA table.



*Figure 5.9 – Shows the CREATE TABLE statements of the IMA\_PRODUCER\_\_SHIPPING\_TO and IMA\_DC\_SHIPPING\_TO tables respectively*

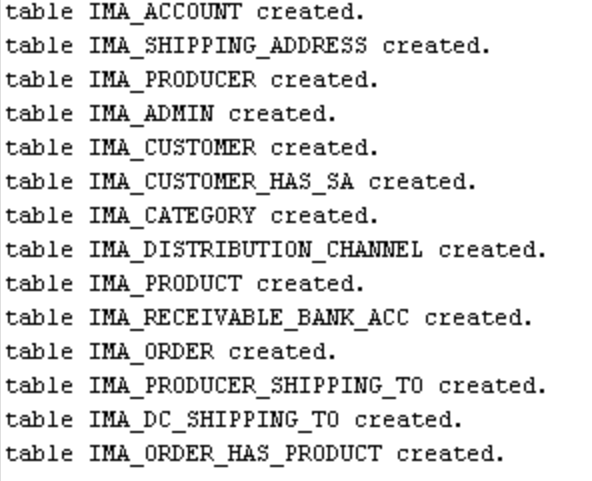
As it can be seen in figure 5.9, the both tables are similarly structured. The only difference is that the IMA\_PRODUCER\_SHIPPING\_TO table references to IMA\_RPODUCER while the IMA\_DC\_SHIPPING\_TO table is related to IMA\_DISTRIBUTION\_CHANNEL. Both of these table are created with a pair of primary keys which all reference to other tables. In other words, these two tables are generated according to the many-to-many relationships between two tables.



*Figure 5.10 – Shows the CREATE TABLE statement of the IMA\_ORDER\_has\_PRODUCT table*

IMA\_ORDER\_has\_PRODUCT is a table which is created based on the relationship between IMA\_ORDER and IMA\_PRODUCT tables. It can be seen in figure 5.10 that its two primary keys reference to those two tables. Moreover, this table contains its own attribute which is Quantity to specify how many items have been ordered for a single type of product.

The SQL scripts shown above have been all executed and successfully committed in SQL Developer as illustrated in the picture below.

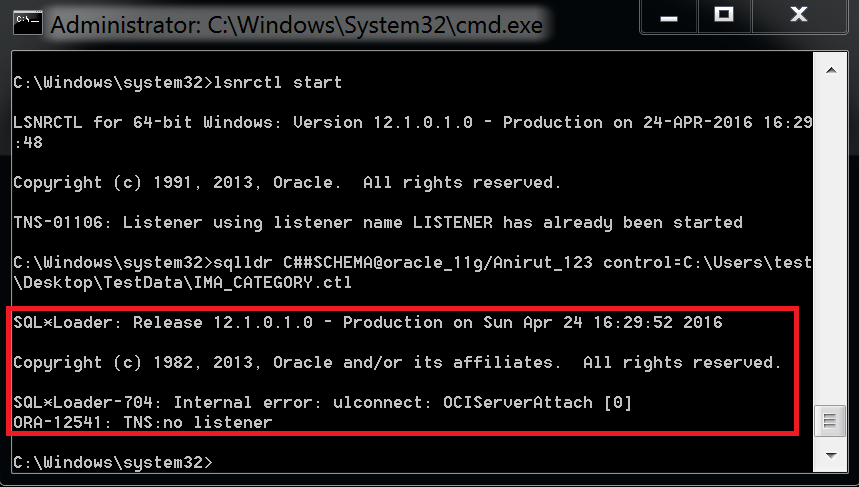


*Figure 5.11 – Shows the result of SQL script execution*

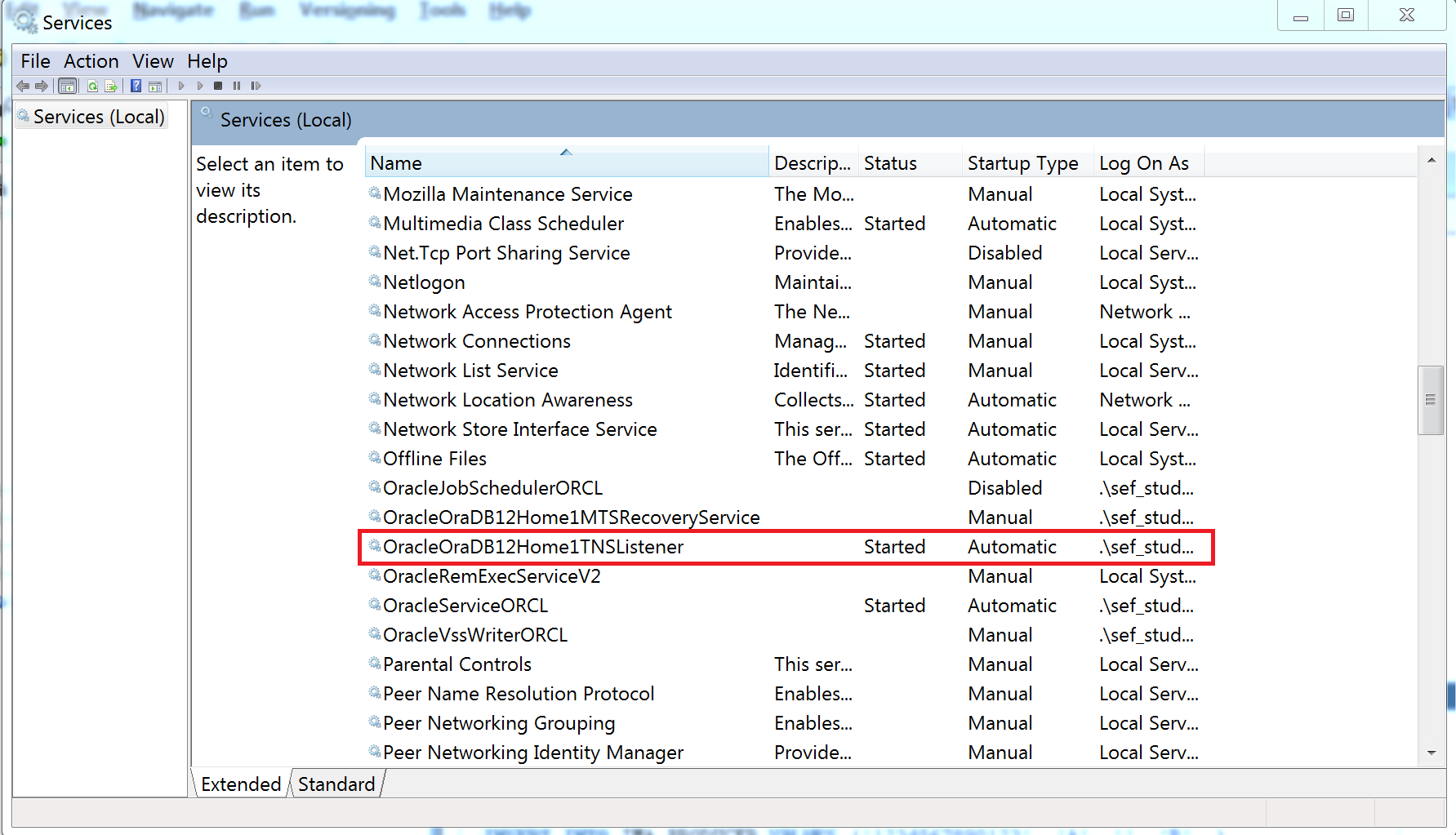
**5.2 Data Preparation**

To investigate whether the database is correctly created, data test must be performed. The data used to test the database is firstly generated based on different scenarios. These data would be imported to the database using SQL\*Loader provided by SQL Plus. Once the test data have been successfully stored in the database, it will be observed in order to perform database normalization.

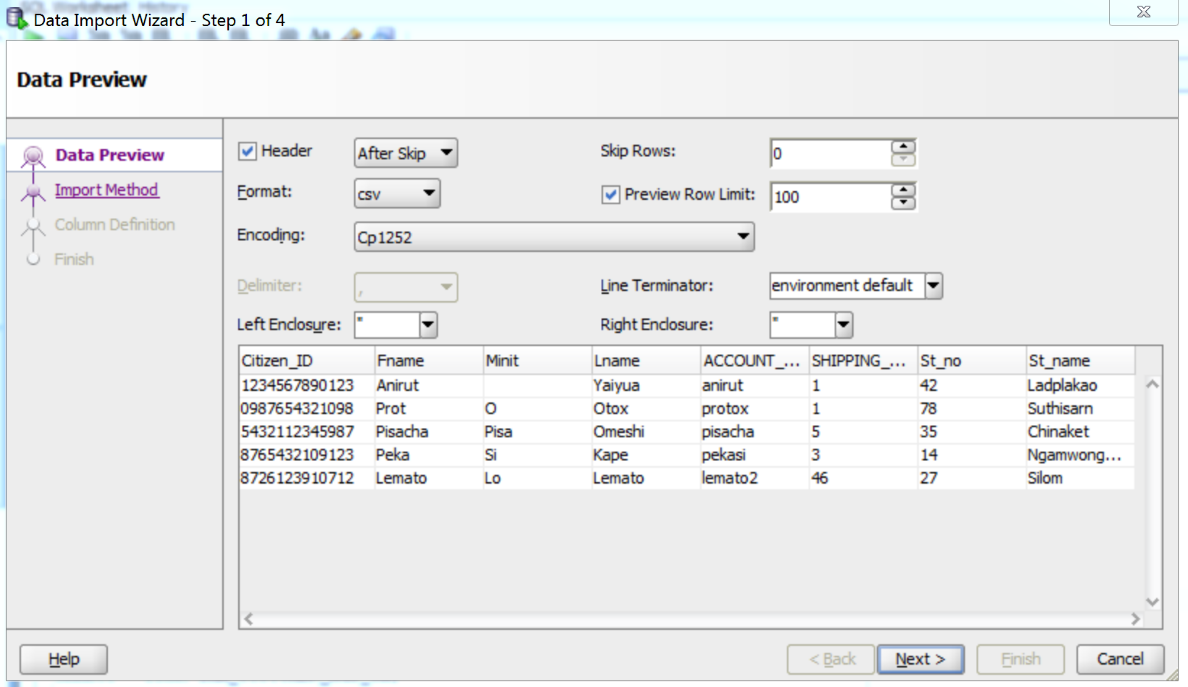
However, SQL\*Loader did not seem to work properly as expected. The error occurred due to a listener issue as shown in figure 5.12. Having seen the SQL\*Loader error, the listener status had been already tested to investigate whether it was running and the result shows that the listener was working properly (see figure 5.13). This issue was also sent to the supervisor but a workaround had also been generated by using data import feature provided in SQL Developer (see an example of data imported into the IMA\_PRODUCER table as shown in figure 5.14). All of the data were imported using this feature.



*Figure 5.12 – Shows there is no listener connected with SQL\*Loader*

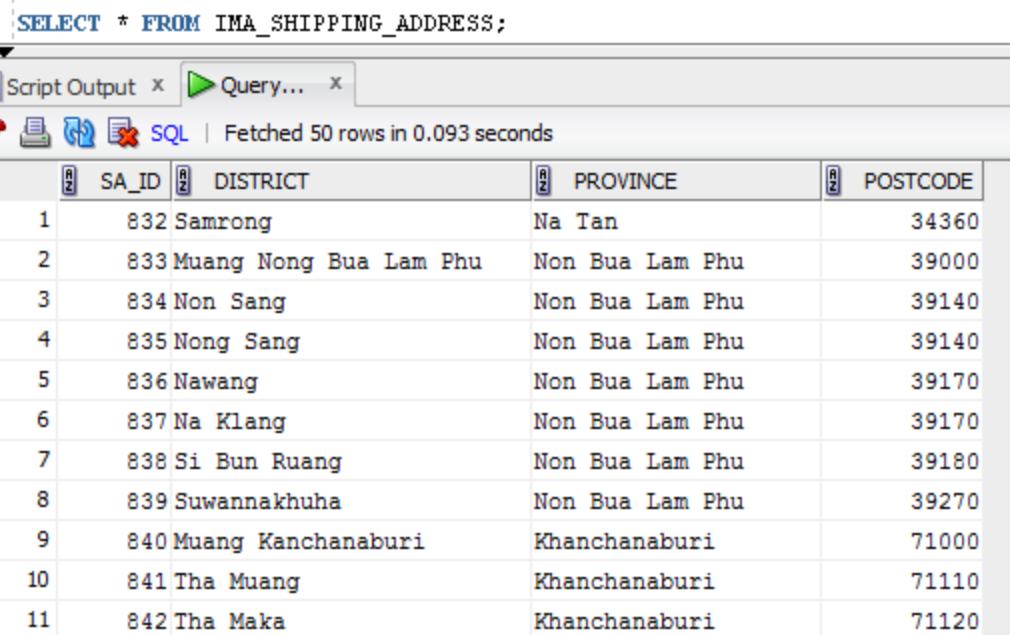
**

*Figure 5.13 – Shows Oracle Listener is started and running properly*

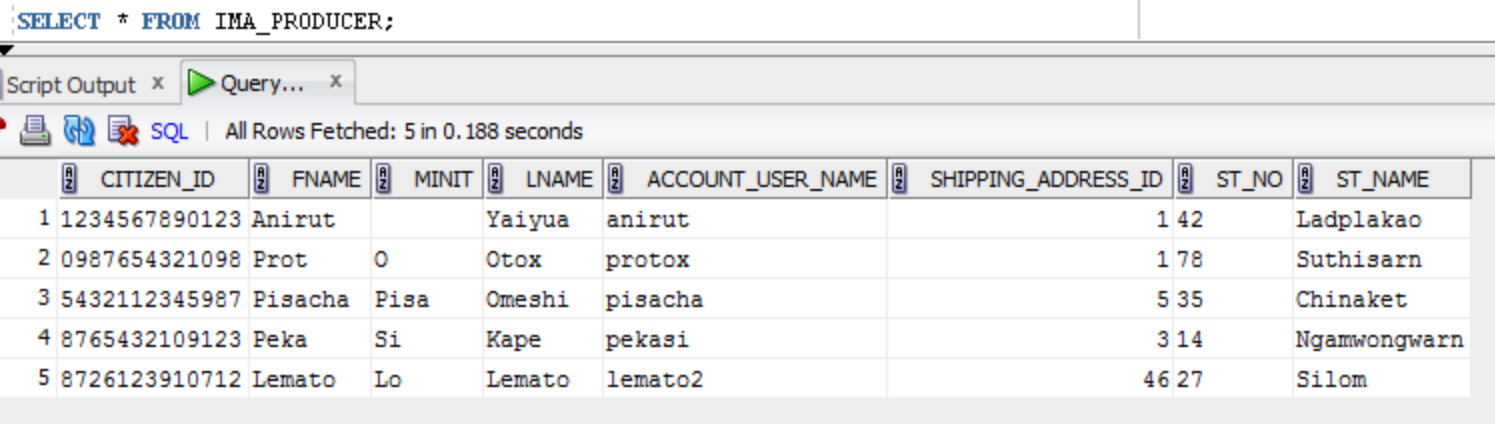


*Figure 5.14 – Shows the data import feature used to store producer data into the IMA\_PRODUCER table*

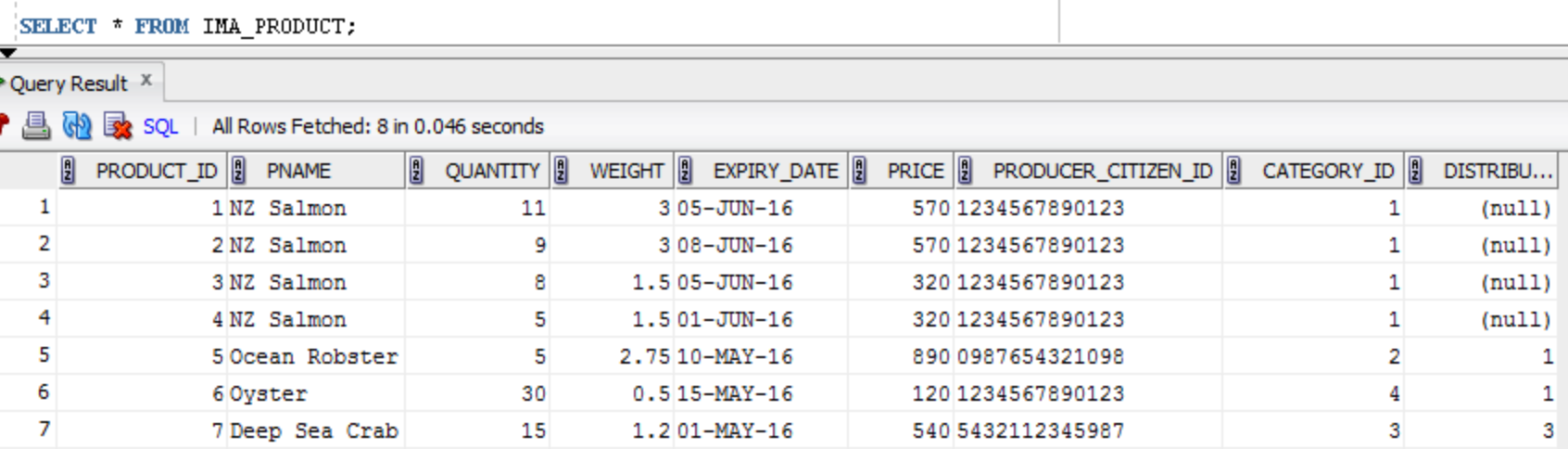
After all data have been successfully imported to the database. It can be tested by executing SQL SELECT command to examine whether they have been correctly imported. As a result, after executing the commands, it shows that all data have been correctly stored in every table. Some of the results are illustrated in figure 5.15, 5.16 and 5.17.



*Figure 5.15 – Shows data of IMA\_SHIPPING\_ADDRESS by executing SQL SELECT command*

**

*Figure 5.16 – Shows data of IMA\_PRODUCER by executing SQL SELECT command*



*Figure 5.17 – Shows data of IMA\_PRODUCT by executing SQL SELECT command*

* 1. **Data Normalization**

It is undeniable that there are several scenarios of data flowing into the database once the application is launched. So it is necessary to ensure that data redundancy is prevented at the start of database implementation. Database normalization is an effective technique used to make the database well-structured and to eliminate data redundancy. It is known that there are various levels to perform the database normalization. In this project, the database normalization will be applied with three levels including first normal form, second normal form and third normal form respectively. The first normal form (1NF) is to ensure that every attribute in a table contains atomic and single value, the 2NF is to make every non-prime attribute dependent on the primary key while the 3NF is to guarantee that there is no non-prim attribute is dependent on other non-prime attribute (Umanath & Scamell, 2007).

After applying the three levels of database normalization, it is found that every table is properly in 1NF, 2NF and 3NF which means that all tables in the database are ready to be storing data without possibility of redundancy to occur.

**References**

* Umanath, N. S., & Scamell, R. W. (2007). *Data modeling and database design*. Boston, Mass;Australia;: Thomson Course Technology.
* Powell, G. (2007). *Oracle 10g database administrator: Implementaton & administration*. Boston, Mass: Thomson Course Technology.