**NITW**

**Computer Science Engineering**

**ONLINE BOOKSTORE MANAGEMENT SYSTEM PROJECT**

By- Snehlata  
 21CSB0B54

**PROBLEM STATEMENT:**

The goal of this project is to design and implement a database for online bookstore management. The database will store information about books, members, reviews, orders, order details and a cart. Additional information may be added.

The database should be able to support the following functionalities:

∙ Store and manage member details and accounts.

∙ Store and retrieve information about books and their reviews.

∙ Allows users to rate and post reviews.

∙ Enable search by subjects and advanced search.

∙ Store and retrieve orders and order details.

∙ Store and retrieve information about books present in the cart.

∙ Purchase books and empty the cart.

∙ Additional functionalities may be added.

The database should be scalable to handle large number of users and a growing amount of data. The database should be designed to ensure data reliability and database consistency. The final solution should be presented in the form of an ER model and a database implementation using a RDBMS such as SQL.

**CONTENTS:**

1) ER model assumptions

2) ER Diagram

3) Primary and Foreign keys representation

4) Functional Dependencies & Normalization

5) Data Dictionary

6) Creation of tables

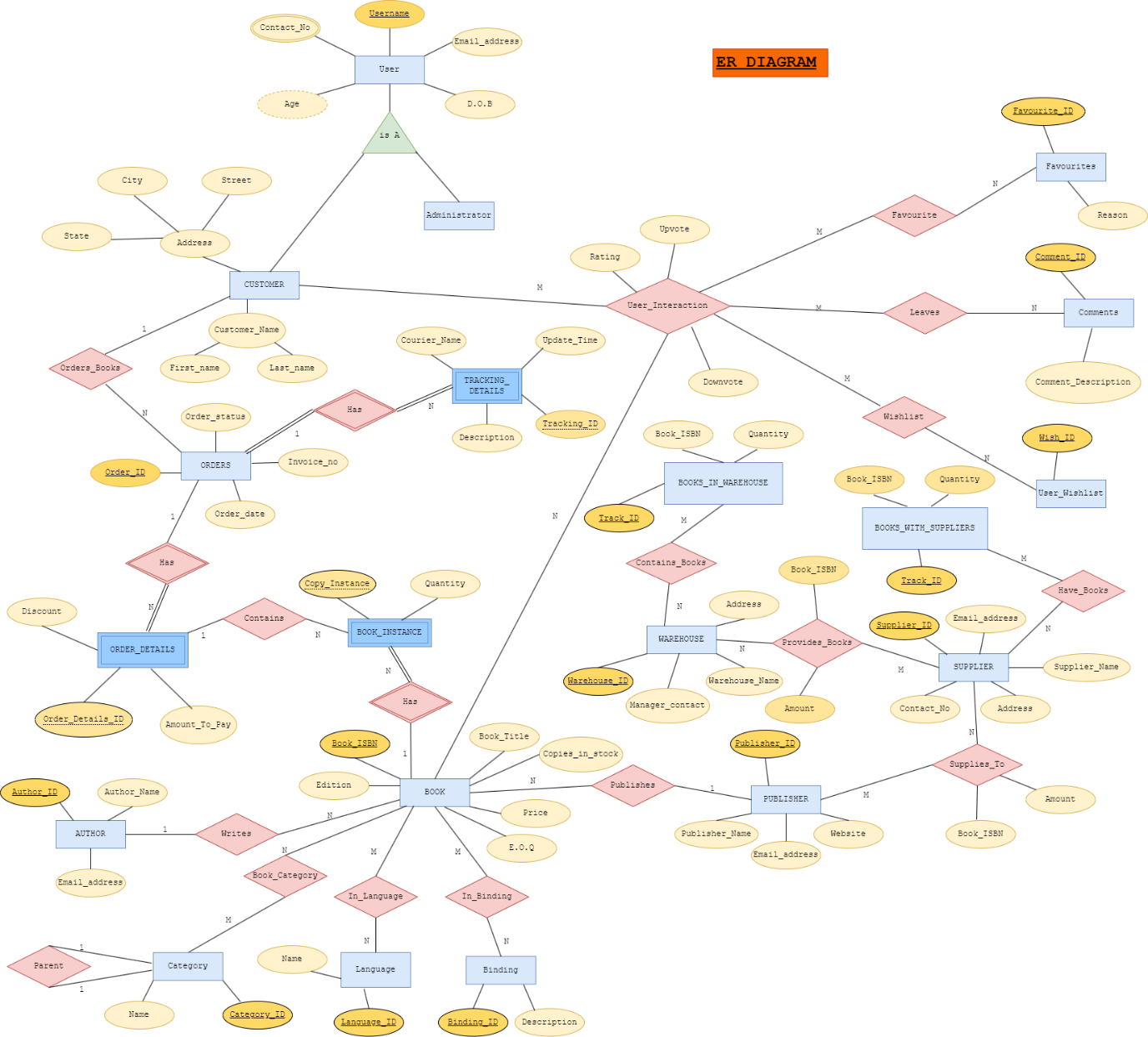
7) Relational Schema

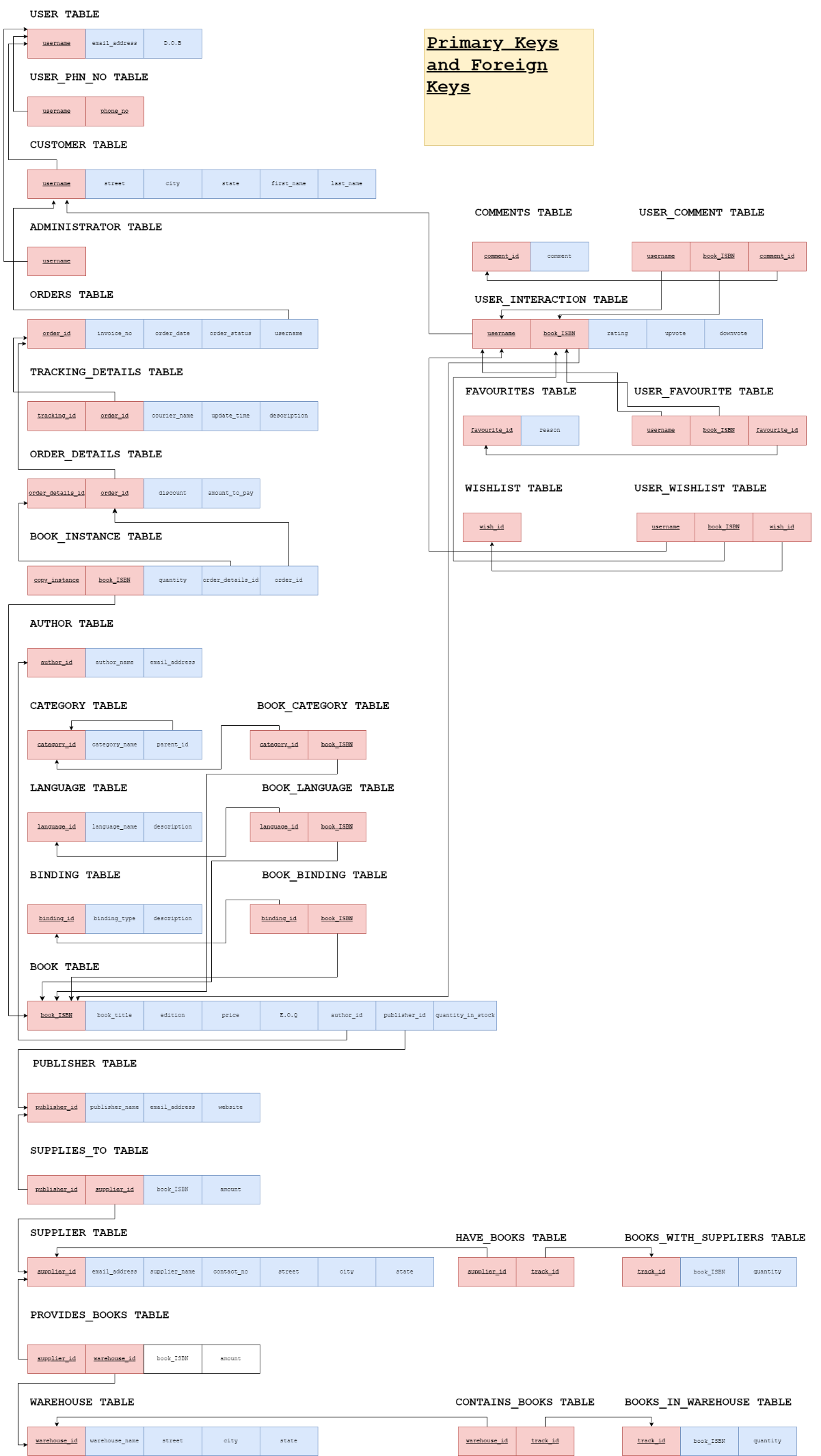
8) Miscellaneous

**1) ER Model Assumptions**

* A return order must return all items in the order, there are no partial returns.
* Not every book is available online, some books are available in the warehouse and suppliers.
* The books are provided from the warehouse to the supplier.
* The supplier then supplies it to the publisher who automatically publishes it to the online web server.
* The books are then ordered by the customers.
* Every detail about the order is handled by the order\_details\_table and book\_instance table.
* The user can interact with the books via rating, upvotes, downvotes, wishlist, favourite the books and leave comments for every book.
* 1 book is written by one author only.

**2)ER Diagram**

****

****

**4)Functional Dependencies and Normalisation**

**USER TABLE**

username -> {email\_address, D.O.B, password, description}

email\_address -> {username, D.O.B, password, description}



The candidate keys are username, email\_address.

Prime attributes = {username, email\_address}

Non prime attributes = {D.O.B, password, description}

There are two functional dependencies.

Both satisfy 1NF as all attributes are atomic.

Both also satisfy 2NF as there are no partial dependencies, no non prime attribute is dependent on any proper subset of the candidate keys. (Candidate keys are atomic)

Both also satisfy 3NF as there are no transitive dependencies. Or we can say that for both the functional dependencies, the LHS is a candidate key, hence it is in 3NF.

Both also satisfy BCNF as the LHS of all dependencies are candidate keys.

**USER\_PHN\_NO TABLE**



Attributes username and phone\_no together form a composite key and there are no other attributes.

Hence it is automatically in BCNF. Since all dependencies are trivial.

**CUSTOMER TABLE**

username -> {street, city, state, first\_name, last\_name}



username is a candidate key.

Prime attributes = {username}

Non prime attributes = {street, city, state, first\_name, last\_name}

There is only one functional dependency.

It satisfies 1NF as all attributes are atomic.

It also satisfies 2NF as there are no partial dependencies, no non prime attribute is dependent on any proper subset of the candidate keys. (Candidate key is atomic)

It also satisfies 3NF as there are no transitive dependencies. Or we can say that for the functional dependency, the LHS is a candidate key, hence it is in 3NF.

It also satisfies BCNF as the LHS of all dependencies are candidate keys.

**ADMINISTRATOR TABLE**

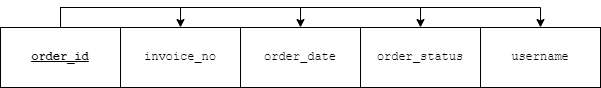
****

There is only one prime attribute and candidate key username.

It is automatically in BCNF because all dependencies are trivial.

**ORDERS TABLE**

order\_id -> {invoice\_no, order\_date, order\_status, username}



order\_id is a candidate key.

Prime attributes = {order\_id}

Non prime attributes = {invoice\_no, order\_date, order\_status, username}

There is only one functional dependency.

It satisfies 1NF as all attributes are atomic.

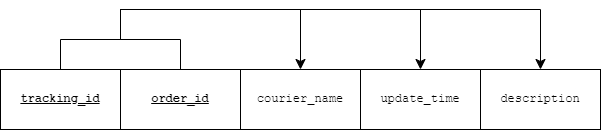
It also satisfies 2NF as there are no partial dependencies, no non prime attribute is dependent on any proper subset of the candidate keys. (Candidate key is atomic)

It also satisfies 3NF as there are no transitive dependencies. Or we can say that for the functional dependency, the LHS is a candidate key, hence it is in 3NF.

It also satisfies BCNF as the LHS of all dependencies are candidate keys.

**TRACKING\_DETAILS TABLE**

tracking\_id order\_id -> {courier\_name, update\_time, description}



tracking\_id order\_id is the candidate key.

Prime attributes = {tracking\_id, order\_id}

Non prime attributes = {courier\_name, update\_time, description}

There is only one functional dependency.

It satisfies 1NF as all attributes are atomic.

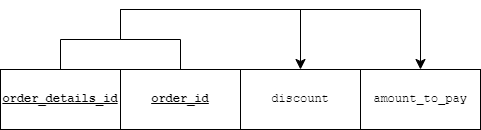
It also satisfies 2NF as there are no partial dependencies, no non prime attribute is dependent on any proper subset of the candidate keys.

It also satisfies 3NF as there are no transitive dependencies. Or we can say that for the functional dependency, the LHS is a candidate key, hence it is in 3NF.

It also satisfies BCNF as the LHS of all dependencies are candidate keys.

**ORDER\_DETAILS TABLE**

order\_details\_id order\_id = {discount, amount\_to\_pay}



order\_details\_id order\_id is the candidate key.

Prime attributes = {order\_details\_id, order\_id}

Non prime attributes = {discount, amount\_to\_pay}

There is only one functional dependency.

It satisfies 1NF as all attributes are atomic.

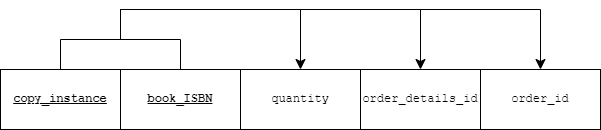
It also satisfies 2NF as there are no partial dependencies, no non prime attribute is dependent on any proper subset of the candidate keys.

It also satisfies 3NF as there are no transitive dependencies. Or we can say that for the functional dependency, the LHS is a candidate key, hence it is in 3NF.

It also satisfies BCNF as the LHS of all dependencies are candidate keys.

**BOOK\_INSTANCE TABLE**

copy\_instance book\_ISBN -> {quantity, order\_details\_id, order\_id}



copy\_instance book\_ISBN is the candidate key.

Prime attributes = {copy\_instance, book\_ISBN}

Non prime attributes = {quantity, order\_details\_id, order\_id}

There is only one functional dependency.

It satisfies 1NF as all attributes are atomic.

It also satisfies 2NF as there are no partial dependencies, no non prime attribute is dependent on any proper subset of the candidate keys.

It also satisfies 3NF as there are no transitive dependencies. Or we can say that for the functional dependency, the LHS is a candidate key, hence it is in 3NF.

It also satisfies BCNF as the LHS of all dependencies are candidate keys.

**AUTHOR TABLE**

author\_id -> {author\_name, email\_address}

email\_address -> {author\_id, author\_name}



author\_id and email\_address are the candidate keys.

Prime attributes = {author\_id, email\_address}

Non prime attributes = {author\_name}

There are two functional dependencies.

Both satisfy 1NF as all attributes are atomic.

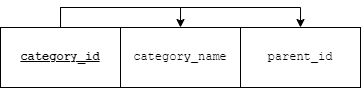
Both also satisfy 2NF as there are no partial dependencies, no non prime attribute is dependent on any proper subset of the candidate keys. (Candidate keys are atomic)

Both also satisfy 3NF as there are no transitive dependencies. Or we can say that for both the functional dependencies, the LHS is a candidate key, hence it is in 3NF.

Both also satisfy BCNF as the LHS of all dependencies are candidate keys.

**CATEGORY TABLE**

category\_id = {category\_name, parent\_id}



category\_id is a candidate key.

Prime attribute = {category\_id}

Non prime attribute = {category\_name, parent\_id}

There is only one functional dependency.

It satisfies 1NF as all attributes are atomic.

It also satisfies 2NF as there are no partial dependencies, no non prime attribute is dependent on any proper subset of the candidate keys.

It also satisfies 3NF as there are no transitive dependencies. Or we can say that for the functional dependency, the LHS is a candidate key, hence it is in 3NF.

It also satisfies BCNF as the LHS of all dependencies are candidate keys.

**BOOK\_CATEGORY TABLE**

****

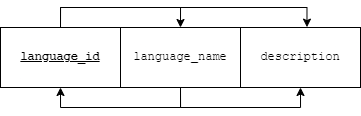
There are only two prime attributes category\_id and book\_ISBN.

It is automatically in BCNF because all dependencies are trivial.

**LANGUAGE TABLE**

language\_id = {language\_name, description}

language\_name = {language\_id, description}



language\_id and language\_name are candidate keys.

Prime attributes = {language\_id, language\_name}

Non prime attributes = {description}

There are two functional dependencies.

Both satisfy 1NF as all attributes are atomic.

Both also satisfy 2NF as there are no partial dependencies, no non prime attribute is dependent on any proper subset of the candidate keys. (Candidate keys are atomic)

Both also satisfy 3NF as there are no transitive dependencies. Or we can say that for both the functional dependencies, the LHS is a candidate key, hence it is in 3NF.

Both also satisfy BCNF as the LHS of all dependencies are candidate keys.

**BOOK\_LANGUAGE TABLE**



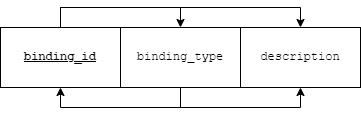
There are only two prime attributes language\_id and book\_ISBN.

It is automatically in BCNF because all dependencies are trivial.

**BINDING TABLE**

binding\_id = {binding \_type, description}

binding \_type = {binding \_id, description}



binding \_id and binding \_type are candidate keys.

Prime attributes = {binding\_id, binding \_type}

Non prime attributes = {description}

There are two functional dependencies.

Both satisfy 1NF as all attributes are atomic.

Both also satisfy 2NF as there are no partial dependencies, no non prime attribute is dependent on any proper subset of the candidate keys. (Candidate keys are atomic)

Both also satisfy 3NF as there are no transitive dependencies. Or we can say that for both the functional dependencies, the LHS is a candidate key, hence it is in 3NF.

Both also satisfy BCNF as the LHS of all dependencies are candidate keys.

**BOOK\_BINDING TABLE**

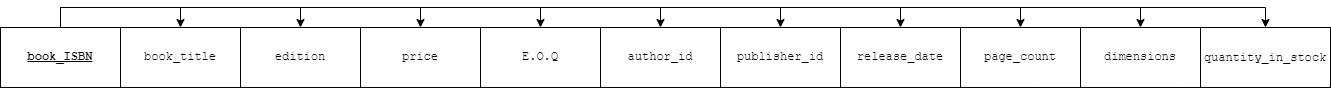


There are only two prime attributes binding\_id and book\_ISBN.

It is automatically in BCNF because all dependencies are trivial.

**BOOK TABLE**

book\_ISBN -> {book\_title, edition, price, E.O.Q, author\_id, publisher\_id, release\_date, page\_count, dimension, quantity\_in\_stock}



book\_ISBN is the candidate key.

Prime attributes = {book\_ISBN}

Non prime attributes = {book\_title, edition, price, E.O.Q, author\_id, publisher\_id, release\_date, page\_count, dimension}

There is only one functional dependency.

It satisfies 1NF as all attributes are atomic.

It also satisfies 2NF as there are no partial dependencies, no non prime attribute is dependent on any proper subset of the candidate keys.

It also satisfies 3NF as there are no transitive dependencies. Or we can say that for the functional dependency, the LHS is a candidate key, hence it is in 3NF.

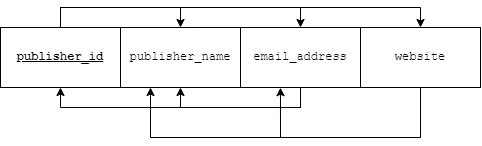
It also satisfies BCNF as the LHS of all dependencies are candidate keys.

**PUBLISHER TABLE**

publisher\_id = {publisher\_name, email\_address, website}

email\_address = {publisher\_name, publisher\_id}

website = {email\_address, publisher\_name}



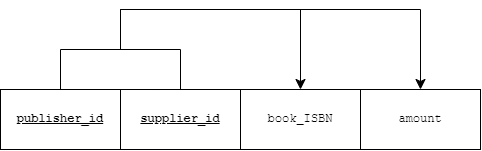
If we check their closures, publisher\_id, email\_address, website are candidate keys.

All three are prime attributes.

It is automatically in BCNF because all dependencies are trivial.

**SUPPLIES\_TO TABLE**

publisher\_id supplier\_id = {book\_ISBN, amount}



publisher\_id supplier\_id is the candidate key.

Prime attributes = {publisher\_id, supplier\_id}

Non prime attributes = {book\_ISBN, amount}

There is only one functional dependency.

It satisfies 1NF as all attributes are atomic.

It also satisfies 2NF as there are no partial dependencies, no non prime attribute is dependent on any proper subset of the candidate keys.

It also satisfies 3NF as there are no transitive dependencies. Or we can say that for the functional dependency, the LHS is a candidate key, hence it is in 3NF.

It also satisfies BCNF as the LHS of all dependencies are candidate keys.

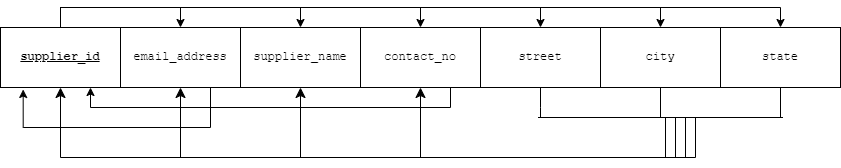
**SUPPLIER TABLE**

supplier\_id = {email\_address, supplier\_name, contact\_no, street, city, state}

street city state = {supplier\_id, email\_address, supplier\_name, contact\_no}

contact\_no = {supplier\_id}

email\_address = {supplier\_id}



supplier\_id, contact\_no, email\_address and street, city, state are candidate keys.

Prime attributes = {supplier\_id, street city state, contact\_no, email\_address}

Non prime attributes = {supplier\_name}

It satisfies 1NF as all attributes are atomic.

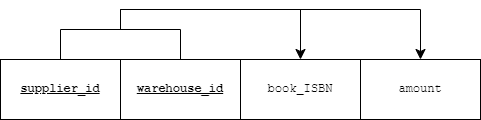
It also satisfies 2NF as there are no partial dependencies, no non prime attribute is dependent on any proper subset of the candidate keys.

It also satisfies 3NF as there are no transitive dependencies. Or we can say that for the functional dependency, the LHS is a candidate key, hence it is in 3NF.

It also satisfies BCNF as the LHS of all dependencies are candidate keys.

**PROVIDES\_BOOKS TABLE**

supplier\_id warehouse\_id = {book\_ISBN, amount}



supplier\_id warehouse\_id is the candidate key.

Prime attributes = {supplier\_id, warehouse\_id}

Non prime attributes = {book\_ISBN, amount}

There is only one functional dependency.

It satisfies 1NF as all attributes are atomic.

It also satisfies 2NF as there are no partial dependencies, no non prime attribute is dependent on any proper subset of the candidate keys.

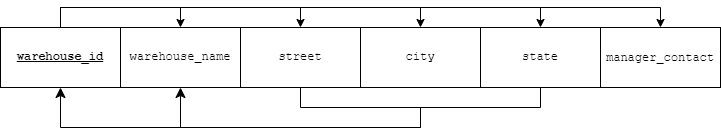
It also satisfies 3NF as there are no transitive dependencies. Or we can say that for the functional dependency, the LHS is a candidate key, hence it is in 3NF.

It also satisfies BCNF as the LHS of all dependencies are candidate keys.

**WAREHOUSE TABLE**

warehouse\_id = {warehouse\_name, street, city, state, manager\_contact}

street city state = {warehouse\_id, warehouse\_name}



warehouse\_id, street city state are the candidate keys.

Prime attributes = {warehouse\_id, street, city, state}

Non prime attributes = {warehouse\_name}

It satisfies 1NF as all attributes are atomic.

It also satisfies 2NF as there are no partial dependencies, no non prime attribute is dependent on any proper subset of the candidate keys.

It also satisfies 3NF as there are no transitive dependencies. Or we can say that for the functional dependency, the LHS is a candidate key, hence it is in 3NF.

It also satisfies BCNF as the LHS of all dependencies are candidate keys.

**USER\_INTERACTION TABLE**

username book\_ISBN = {rating, upvote, downvote}



username book\_ISBN is the candidate key.

Prime attributes = {username, book\_ISBN}

Non prime attributes = {rating, upvote, downvote}

There is only one functional dependency.

It satisfies 1NF as all attributes are atomic.

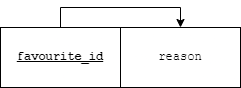
It also satisfies 2NF as there are no partial dependencies, no non prime attribute is dependent on any proper subset of the candidate keys.

It also satisfies 3NF as there are no transitive dependencies. Or we can say that for the functional dependency, the LHS is a candidate key, hence it is in 3NF.

It also satisfies BCNF as the LHS of all dependencies are candidate keys.

**FAVOURITES TABLE**

favourite\_id = {reason}



favourite\_id is the candidate key.

It is automatically in BCNF as the LHS are all candidate keys.

**USER\_FAVOURITE TABLE**

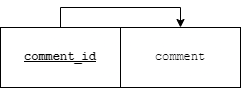


All are prime attributes.

It is automatically in BCNF as all dependencies are trivial.

**COMMENTS TABLE**

comment\_id = {comment}



comment\_id is the candidate key.

It is automatically in BCNF as the LHS are all candidate keys.

**USER\_COMMENT TABLE**



All are prime attributes.

It is automatically in BCNF as all dependencies are trivial.

**WISHLIST\_TABLE**



There is only one prime attribute.

It is automatically in BCNF as all dependencies are trivial.

**USER\_WISHLIST TABLE**



All are prime attributes.

It is automatically in BCNF as all dependencies are trivial.

**HAVE\_BOOKS TABLE**

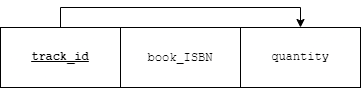
****

All are prime attributes.

It is automatically in BCNF as all dependencies are trivial.

**BOOKS\_WITH\_SUPPLIERS TABLE**

track\_id = {book\_ISBN, quantity}



track\_id is the candidate key.

It is automatically in BCNF as the LHS are all candidate keys.

**CONTAINS\_BOOKS TABLE**

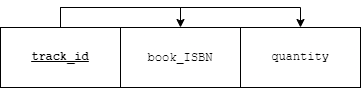
****

All are prime attributes.

It is automatically in BCNF as all dependencies are trivial.

**BOOKS\_IN\_WAREHOUSE TABLE**

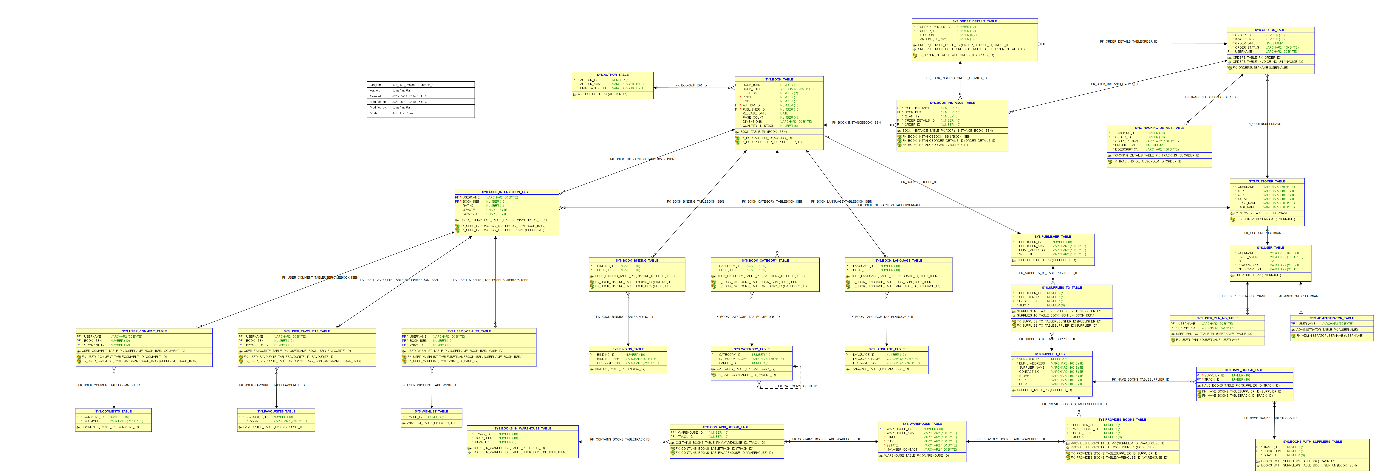
track\_id = {book\_ISBN, quantity}



track\_id is the candidate key.

It is automatically in BCNF as the LHS are all candidate keys.

**7)Relational Schema**

****

**5)Data Dictionary**

1. **Data Dictionary for USER**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column Status | Attribute | Data type | Size | Description |
| Not Null  Primary Key | username | VARCHAR2 | 50 | It is a unique way of identifying every user. |
|  | email\_address | VARCHAR2 | 100 | Address associated with a user. |
|  | D.O.B | DATE |  | Date of birth of the user. |
| Not Null | password | VARCHAR2 | 20 | Password typed in by the user. |
|  | description | VARCHAR2 | 255 | About section of the user. Can be null. |

1. **Data Dictionary for USER\_PHN\_NO**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column Status | Attribute | Data type | Size | Description |
| Not Null  Composite Key  Foreign Key | username | VARCHAR2 | 50 |  |
| Not Null  Composite Key | phone\_no | VARCHAR2 | 100 | Phone number associated with the user. |

1. **Data Dictionary for CUSTOMER**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column Status | Attribute | Data type | Size | Description |
| Not Null  Primary Key  Foreign Key | username | VARCHAR2 | 50 |  |
| Not Null | street | VARCHAR2 | 100 | Address. |
| Not Null | city | VARCHAR2 | 100 | Address. |
| Not Null | state | VARCHAR2 | 100 | Address. |
|  | first\_name | VARCHAR2 | 50 | Name of the user. |
|  | last\_name | VARCHAR2 | 50 | Name of the user. |

1. **Data Dictionary for ADMINISTRATOR**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column Status | Attribute | Data type | Size | Description |
| Not Null  Primary Key  Foreign Key | username | VARCHAR2 | 50 |  |

1. **Data Dictionary for ORDERS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column Status | Attribute | Data type | Size | Description |
| Not Null  Primary Key | order\_id | NUMBER | 10, 0 | Unique ID for each order. |
| Not Null  Unique | invoice\_no | NUMBER | 10, 0 | Unique ID between client and the seller. |
| Not Null | order\_date | TIMESTAMP |  | Date issued for the order. |
| Not Null | order\_status | VARCHAR2 | 100 | Delivered or not. |
| Not Null  Foreign Key | username | VARCHAR2 | 50 |  |

1. **Data Dictionary for TRACKING\_DETAILS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column Status | Attribute | Data type | Size | Description |
| Not Null  Composite Key | tracking\_id | NUMBER | 10, 0 | Unique ID for tracking. |
| Not Null  Composite Key  Foreign Key | order\_id | NUMBER | 10, 0 |  |
| Not Null | courier\_name | VARCHAR2 | 100 | Name of the deliverer, for contact. |
| Not Null | update\_time | TIMESTAMP |  | Time updated. |
| Not Null | description | VARCHAR2 | 100 | Cannot be null. |

1. **Data Dictionary for ORDER\_DETAILS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column Status | Attribute | Data type | Size | Description |
| Not Null  Composite Key  Unique | order\_details\_id | NUMBER | 10, 0 | Unique ID for details. |
| Not Null  Composite Key  Foreign Key | order\_id | NUMBER | 10, 0 |  |
| Not Null | discount | DECIMAL | 5, 2 | Discount, can be between 0 and 100. |
| Not Null | amount\_to\_pay | NUMBER | 10, 3 | Amount to be paid |

1. **Data Dictionary for BOOK\_INSTANCE**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column Status | Attribute | Data type | Size | Description |
| Not Null  Composite Key | copy\_instance | NUMBER | 10, 0 | Copy instance of the book. |
| Not Null  Composite Key  Foreign Key | book\_ISBN | NUMBER | 10, 0 | Unique ID for each book. |
| Not Null | quantity | NUMBER | 5, 0 | Quantity in each order. |
| Not Null  Foreign Key | order\_details\_id | NUMBER | 10, 0 |  |
| Not Null  Foreign Key | order\_id | NUMBER | 10, 0 |  |

1. **Data Dictionary for AUTHOR**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column Status | Attribute | Data type | Size | Description |
| Not Null  Primary Key | author\_id | NUMBER | 10, 0 | Unique ID for each author. |
| Not Null | author\_name | VARCHAR2 | 50 | Name of the author. |
| Not Null | email\_address | VARCHAR2 | 100 | Email address of the author |

1. **Data Dictionary for CATEGORY**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column Status | Attribute | Data type | Size | Description |
| Not Null  Primary Key | category\_id | NUMBER | 10, 0 | ID of the category. |
| Not Null | category\_name | VARCHAR2 | 50 | Name of the category. |
| Foreign Key | parent\_id | NUMBER | 10, 0 | Subgenre. |

1. **Data Dictionary for LANGUAGE**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column Status | Attribute | Data type | Size | Description |
| Not Null  Primary Key | language\_id | NUMBER | 10, 0 | ID of the language. |
| Not Null | language\_name | VARCHAR2 | 50 | Name of the language. |
|  | description | VARCHAR2 | 255 | Can be NULL. |

1. **Data Dictionary for BINDING**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column Status | Attribute | Data type | Size | Description |
| Not Null  Primary Key | binding\_id | NUMBER | 10, 0 | ID of the binding. |
| Not Null | binding\_type | VARCHAR2 | 50 | Name of the binding. |
|  | description | VARCHAR2 | 255 |  |

1. **Data Dictionary for BOOK\_CATEGORY**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column Status | Attribute | Data type | Size | Description |
| Not Null  Composite Key  Foreign Key | category\_id | NUMBER | 10, 0 |  |
| Not Null  Composite Key  Foreign Key | book\_ISBN | NUMBER | 10, 0 |  |

1. **Data Dictionary for BOOK\_LANGUAGE**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column Status | Attribute | Data type | Size | Description |
| Not Null  Composite Key  Foreign Key | language\_id | NUMBER | 10, 0 |  |
| Not Null  Composite Key  Foreign Key | book\_ISBN | NUMBER | 10, 0 |  |

1. **Data Dictionary for BOOK\_BINDING**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column Status | Attribute | Data type | Size | Description |
| Not Null  Composite Key  Foreign Key | binding\_id | NUMBER | 10, 0 |  |
| Not Null  Composite Key  Foreign Key | book\_ISBN | NUMBER | 10, 0 |  |

1. **Data Dictionary for BOOK**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column Status | Attribute | Data type | Size | Description |
| Not Null  Primary Key | book\_ISBN | NUMBER | 10, 0 |  |
|  | book\_title | VARCHAR2 | 255 | Title of the book. |
|  | edition | NUMBER | 2, 0 | Edition number. |
| Not Null | price | NUMBER | 7, 3 | Price of the book. |
|  | E.O.Q | NUMBER | 7, 3 | EOQ |
| Not Null  Foreign Key | author\_id | NUMBER | 10, 0 |  |
| Not Null  Foreign Key | publisher\_id | NUMBER | 10, 0 |  |
|  | release\_date | DATE |  |  |
|  | page\_count | NUMBER | 4, 0 |  |
|  | dimensions | VARCHAR2 | 50 |  |
| Not Null | quantity\_in\_stock | NUMBER | 6, 0 |  |

1. **Data Dictionary for PUBLISHER**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column Status | Attribute | Data type | Size | Description |
| Not Null  Primary Key | publisher\_id | NUMBER | 10, 0 | Publisher ID |
| Not Null | publisher\_name | VARCHAR2 | 100 |  |
| Not Null | email\_address | VARCHAR2 | 100 |  |
| Not Null | website | VARCHAR2 | 100 |  |

1. **Data Dictionary for SUPPLIES\_TO**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column Status | Attribute | Data type | Size | Description |
| Not Null  Composite Key  Foreign Key | publisher\_id | NUMBER | 10, 0 |  |
| Not Null  Composite Key  Foreign Key | supplier\_id | NUMBER | 10, 0 | Supplier ID. |
| Not Null  Unique | book\_ISBN | NUMBER | 10, 0 |  |
| Not Null | amount | NUMBER | 6, 0 |  |

1. **Data Dictionary for SUPPLIER**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column Status | Attribute | Data type | Size | Description |
| Not Null  Primary Key | supplier\_id | NUMBER | 10, 0 |  |
| Not Null | email\_address | VARCHAR2 | 100 | Email of the supplier. |
| Not Null | supplier\_name | VARCHAR2 | 100 |  |
| Not Null | contact\_no | VARCHAR2 | 100 | Contact number. |
| Not Null | street | VARCHAR2 | 100 |  |
| Not Null | city | VARCHAR2 | 100 |  |
| Not Null | state | VARCHAR2 | 100 |  |

1. **Data Dictionary for PROVIDES\_BOOKS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column Status | Attribute | Data type | Size | Description |
| Not Null  Composite Key  Foreign Key | supplier\_id | NUMBER | 10, 0 |  |
| Not Null  Composite Key  Foreign Key | warehouse\_id | NUMBER | 10, 0 |  |
| Not Null  Unique | book\_ISBN | NUMBER | 10, 0 |  |
| Not Null | amount | NUMBER | 6, 0 | Amount of books. |

1. **Data Dictionary for WAREHOUSE**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column Status | Attribute | Data type | Size | Description |
| Not Null  Primary Key | warehouse\_id | NUMBER | 10, 0 |  |
| Not Null | warehouse\_name | VARCHAR2 | 100 |  |
| Not Null | street | VARCHAR2 | 100 |  |
| Not Null | city | VARCHAR2 | 100 |  |
| Not Null | state | VARCHAR2 | 100 |  |
| Not Null | manager\_contact | VARCHAR2 | 100 |  |

1. **Data Dictionary for HAVE\_BOOKS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column Status | Attribute | Data type | Size | Description |
| Not Null  Composite Key  Foreign Key | supplier\_id | NUMBER | 10, 0 |  |
| Not Null  Composite Key  Foreign Key | track\_id | NUMBER | 10, 0 |  |

1. **Data Dictionary for BOOKS\_WITH\_SUPPLIERS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column Status | Attribute | Data type | Size | Description |
| Not Null  Primary Key | track\_id | NUMBER | 10, 0 |  |
| Not Null  Unique | book\_ISBN | NUMBER | 10, 0 |  |
| Not Null | quantity | NUMBER | 6, 0 |  |

1. **Data Dictionary for CONTAINS\_BOOKS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column Status | Attribute | Data type | Size | Description |
| Not Null  Composite Key  Foreign Key | warehouse\_id | NUMBER | 10, 0 |  |
| Not Null  Composite Key  Foreign Key | track\_id | NUMBER | 10, 0 |  |

1. **Data Dictionary for BOOKS\_IN\_WAREHOUSE**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column Status | Attribute | Data type | Size | Description |
| Not Null  Primary Key | track\_id | NUMBER | 10, 0 |  |
| Not Null  Unique | book\_ISBN | NUMBER | 10, 0 |  |
| Not Null | quantity | NUMBER | 6, 0 |  |

1. **Data Dictionary for USER\_INTERACTION**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column Status | Attribute | Data type | Size | Description |
| Not Null  Composite Key  Foreign Key | username | VARCHAR2 | 50 |  |
| Not Null  Composite Key  Foreign Key | book\_ISBN | NUMBER | 10, 0 |  |
|  | rating | NUMBER | 1, 0 |  |
|  | upvote | CHAR | 1 |  |
|  | downvote | CHAR | 1 |  |

1. **Data Dictionary for FAVOURITES**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column Status | Attribute | Data type | Size | Description |
| Not Null  Primary Key | favourite\_id | NUMBER | 10, 0 |  |
|  | reason | VARCHAR2 | 255 |  |

1. **Data Dictionary for USER\_FAVOURITE**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column Status | Attribute | Data type | Size | Description |
| Not Null  Composite Key  Foreign Key | username | VARCHAR2 | 50 |  |
| Not Null  Composite Key  Foreign Key | book\_ISBN | NUMBER | 10, 0 |  |
| Not Null  Composite Key  Foreign Key | favourite\_id | NUMBER | 10, 0 |  |

1. **Data Dictionary for COMMENTS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column Status | Attribute | Data type | Size | Description |
| Not Null  Primary Key | comment\_id | NUMBER | 10, 0 |  |
| Not Null | comment | VARCHAR2 | 255 |  |

1. **Data Dictionary for USER\_COMMENT**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column Status | Attribute | Data type | Size | Description |
| Not Null  Composite Key  Foreign Key | username | VARCHAR2 | 50 |  |
| Not Null  Composite Key  Foreign Key | book\_ISBN | NUMBER | 10, 0 |  |
| Not Null  Composite Key  Foreign Key | comment\_id | NUMBER | 10, 0 |  |

1. **Data Dictionary for WISHLIST**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column Status | Attribute | Data type | Size | Description |
| Not Null  Primary Key | wish\_id | NUMBER | 10, 0 |  |

1. **Data Dictionary for USER\_WISHLIST**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column Status | Attribute | Data type | Size | Description |
| Not Null  Composite Key  Foreign Key | username | VARCHAR2 | 50 |  |
| Not Null  Composite Key  Foreign Key | book\_ISBN | NUMBER | 10, 0 |  |
| Not Null  Composite Key  Foreign Key | wish\_id | NUMBER | 10, 0 |  |

**6)Creation of tables**

**USER\_TABLE**

CREATE TABLE USER\_TABLE

(

username VARCHAR2(50) NOT NULL PRIMARY KEY,

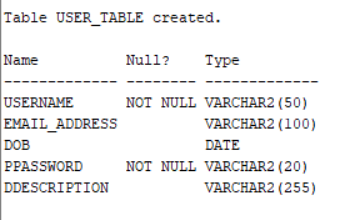
email\_address VARCHAR2(100),

DOB DATE,

ppassword VARCHAR2(20) NOT NULL,

ddescription VARCHAR2(255)

);



**USER\_PHN\_NO\_TABLE**

CREATE TABLE USER\_PHN\_NO\_TABLE

(

username VARCHAR2(50) NOT NULL,

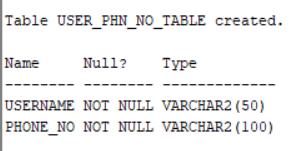
phone\_no VARCHAR2(100) NOT NULL,

PRIMARY KEY(username, phone\_no),

CONSTRAINT FK\_USER\_PHN\_NOusername FOREIGN KEY

(username) REFERENCES USER\_TABLE (username) ON DELETE CASCADE

);



**CUSTOMER\_TABLE**

CREATE TABLE CUSTOMER\_TABLE

(

username VARCHAR2(50) NOT NULL PRIMARY KEY,

street VARCHAR2(100) NOT NULL,

city VARCHAR2(100) NOT NULL,

sstate VARCHAR2(100) NOT NULL,

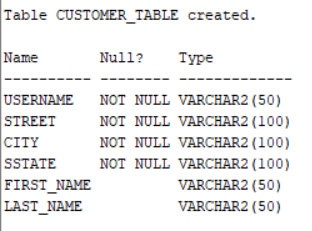
first\_name VARCHAR2(50),

last\_name VARCHAR2(50),

CONSTRAINT FK\_CUSTOMERusername FOREIGN KEY

(username) REFERENCES USER\_TABLE (username) ON DELETE CASCADE

);



**ADMINISTRATOR\_TABLE**

CREATE TABLE ADMINISTRATOR\_TABLE

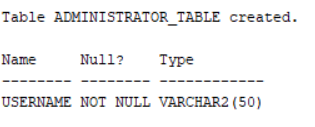
(

username VARCHAR2(50) NOT NULL PRIMARY KEY,

CONSTRAINT FK\_ADMINISTRATORusername FOREIGN KEY

(username) REFERENCES USER\_TABLE (username) ON DELETE CASCADE

);



**ORDERS\_TABLE**

CREATE TABLE ORDERS\_TABLE

(

order\_id NUMBER(10, 0) NOT NULL PRIMARY KEY,

invoice\_no NUMBER(10, 0) NOT NULL UNIQUE,

order\_date TIMESTAMP DEFAULT CURRENT\_DATE NOT NULL,

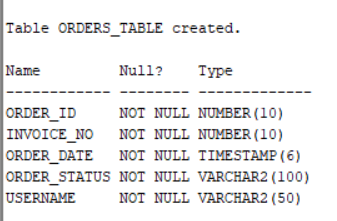
order\_status VARCHAR2(100) NOT NULL,

username VARCHAR2(50) NOT NULL,

CONSTRAINT FK\_ORDERSusername FOREIGN KEY

(username) REFERENCES CUSTOMER\_TABLE (username) ON DELETE CASCADE

);



**TRACKING\_DETAILS\_TABLE**

CREATE TABLE TRACKING\_DETAILS\_TABLE

(

tracking\_id NUMBER(10, 0) NOT NULL,

order\_id NUMBER(10, 0) NOT NULL,

courier\_name VARCHAR2(100) NOT NULL,

update\_time TIMESTAMP DEFAULT CURRENT\_DATE NOT NULL,

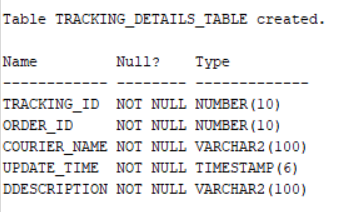
ddescription VARCHAR2(100) NOT NULL,

PRIMARY KEY(tracking\_id, order\_id),

CONSTRAINT FK\_TRACKING\_DETAILSorder\_id FOREIGN KEY

(order\_id) REFERENCES ORDERS\_TABLE (order\_id) ON DELETE CASCADE

);



**ORDER\_DETAILS\_TABLE**

CREATE TABLE ORDER\_DETAILS\_TABLE

(

order\_details\_id NUMBER(10, 0) NOT NULL UNIQUE,

order\_id NUMBER(10, 0) NOT NULL,

discount DECIMAL(5, 2) DEFAULT 0 CHECK (discount >= 0 AND discount <= 100) NOT NULL,

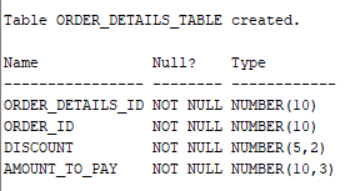
amount\_to\_pay NUMBER(10, 3) DEFAULT 0 NOT NULL,

PRIMARY KEY(order\_details\_id, order\_id),

CONSTRAINT FK\_ORDER\_DETAILS\_TABLEorder\_id FOREIGN KEY

(order\_id) REFERENCES ORDERS\_TABLE (order\_id) ON DELETE CASCADE

);



**AUTHOR\_TABLE**

CREATE TABLE AUTHOR\_TABLE

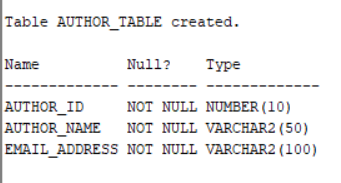
(

author\_id NUMBER(10, 0) NOT NULL PRIMARY KEY,

author\_name VARCHAR2(50) NOT NULL,

email\_address VARCHAR2(100) NOT NULL

);



**PUBLISHER\_TABLE**

CREATE TABLE PUBLISHER\_TABLE

(

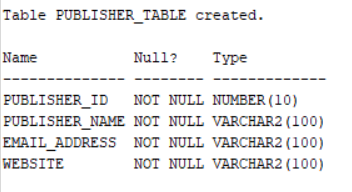
publisher\_id NUMBER(10, 0) NOT NULL PRIMARY KEY,

publisher\_name VARCHAR2(100) NOT NULL,

email\_address VARCHAR2(100) NOT NULL,

website VARCHAR2(100) NOT NULL

);



**BOOK\_TABLE**

CREATE TABLE BOOK\_TABLE

(

book\_ISBN NUMBER(10, 0) NOT NULL PRIMARY KEY,

book\_title VARCHAR2(255),

eedition NUMBER(2, 0),

price NUMBER(7, 3) NOT NULL,

EOQ NUMBER(7, 3),

author\_id NUMBER(10, 0) NOT NULL,

publisher\_id NUMBER(10, 0) NOT NULL,

release\_date DATE,

page\_count NUMBER(4, 0),

dimensions VARCHAR2(50),

quantity\_in\_stock NUMBER(6, 0) DEFAULT 0 CHECK (quantity\_in\_stock >= 0) NOT NULL,

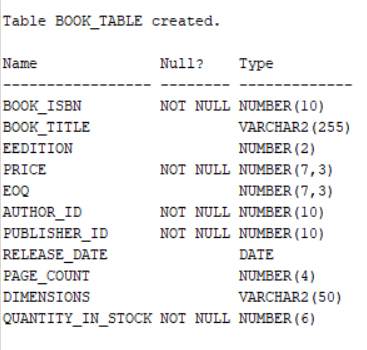
CONSTRAINT FK\_BOOKauthor\_id FOREIGN KEY

(author\_id) REFERENCES AUTHOR\_TABLE (author\_id) ON DELETE CASCADE,

CONSTRAINT FK\_BOOKpublisher\_id FOREIGN KEY

(publisher\_id) REFERENCES PUBLISHER\_TABLE (publisher\_id) ON DELETE CASCADE

);



**BOOK\_INSTANCE\_TABLE**

CREATE TABLE BOOK\_INSTANCE\_TABLE

(

copy\_instance NUMBER(10, 0) NOT NULL,

book\_ISBN NUMBER(10, 0) NOT NULL,

quantity NUMBER(5, 0) NOT NULL,

order\_details\_id NUMBER(10, 0) NOT NULL,

order\_id NUMBER(10, 0) NOT NULL,

PRIMARY KEY(copy\_instance, book\_ISBN),

CONSTRAINT FK\_BOOK\_INSTANCEbook\_ISBN FOREIGN KEY

(book\_ISBN) REFERENCES BOOK\_TABLE (book\_ISBN) ON DELETE CASCADE,

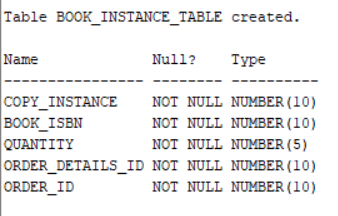
CONSTRAINT FK\_BOOK\_INSTANCEorder\_details\_id FOREIGN KEY

(order\_details\_id) REFERENCES ORDER\_DETAILS\_TABLE (order\_details\_id) ON DELETE CASCADE,

CONSTRAINT FK\_BOOK\_INSTANCEorder\_id FOREIGN KEY

(order\_id) REFERENCES ORDERS\_TABLE (order\_id) ON DELETE CASCADE

);



**CATEGORY\_TABLE**

CREATE TABLE CATEGORY\_TABLE

(

category\_id NUMBER(10, 0) NOT NULL PRIMARY KEY,

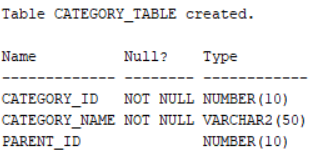
category\_name VARCHAR2(50) NOT NULL,

parent\_id NUMBER(10, 0),

CONSTRAINT FK\_CATEGORYparent\_id FOREIGN KEY

(parent\_id) REFERENCES CATEGORY\_TABLE (category\_id) ON DELETE SET NULL

);



**LANGUAGE\_TABLE**

CREATE TABLE LANGUAGE\_TABLE

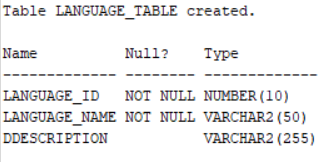
(

language\_id NUMBER(10, 0) NOT NULL PRIMARY KEY,

language\_name VARCHAR2(50) NOT NULL,

ddescription VARCHAR2(255)

);



**BINDING\_TABLE**

CREATE TABLE BINDING\_TABLE

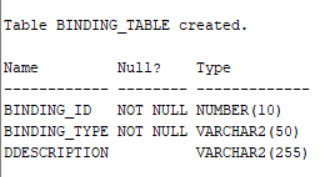
(

binding\_id NUMBER(10, 0) NOT NULL PRIMARY KEY,

binding\_type VARCHAR2(50) NOT NULL,

ddescription VARCHAR2(255)

);



**BOOK\_CATEGORY\_TABLE**

CREATE TABLE BOOK\_CATEGORY\_TABLE

(

category\_id NUMBER(10, 0) NOT NULL,

book\_ISBN NUMBER(10, 0) NOT NULL,

PRIMARY KEY(category\_id, book\_ISBN),

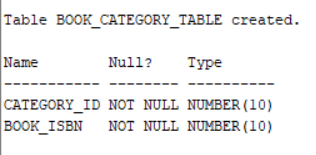
CONSTRAINT FK\_BOOK\_CATEGORY\_TABLEcategory\_id FOREIGN KEY

(category\_id) REFERENCES CATEGORY\_TABLE (category\_id) ON DELETE CASCADE,

CONSTRAINT FK\_BOOK\_CATEGORY\_TABLEbook\_ISBN FOREIGN KEY

(book\_ISBN) REFERENCES BOOK\_TABLE (book\_ISBN) ON DELETE CASCADE

);



**BOOK\_LANGUAGE\_TABLE**

CREATE TABLE BOOK\_LANGUAGE\_TABLE

(

language\_id NUMBER(10, 0) NOT NULL,

book\_ISBN NUMBER(10, 0) NOT NULL,

PRIMARY KEY(language\_id, book\_ISBN),

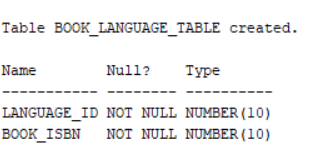
CONSTRAINT FK\_BOOK\_LANGUAGE\_TABLElanguage\_id FOREIGN KEY

(language\_id) REFERENCES LANGUAGE\_TABLE (language\_id) ON DELETE CASCADE,

CONSTRAINT FK\_BOOK\_LANGUAGE\_TABLEbook\_ISBN FOREIGN KEY

(book\_ISBN) REFERENCES BOOK\_TABLE (book\_ISBN) ON DELETE CASCADE

);



**BOOK\_BINDING\_TABLE**

CREATE TABLE BOOK\_BINDING\_TABLE

(

binding\_id NUMBER(10, 0) NOT NULL,

book\_ISBN NUMBER(10, 0) NOT NULL,

PRIMARY KEY(binding\_id, book\_ISBN),

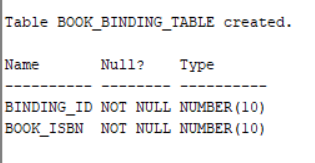
CONSTRAINT FK\_BOOK\_BINDING\_TABLEbinding\_id FOREIGN KEY

(binding\_id) REFERENCES BINDING\_TABLE (binding\_id) ON DELETE CASCADE,

CONSTRAINT FK\_BOOK\_BINDING\_TABLEbook\_ISBN FOREIGN KEY

(book\_ISBN) REFERENCES BOOK\_TABLE (book\_ISBN) ON DELETE CASCADE

);



**SUPPLIER\_TABLE**

CREATE TABLE SUPPLIER\_TABLE

(

supplier\_id NUMBER(10, 0) NOT NULL PRIMARY KEY,

email\_address VARCHAR2(100) NOT NULL,

supplier\_name VARCHAR2(100) NOT NULL,

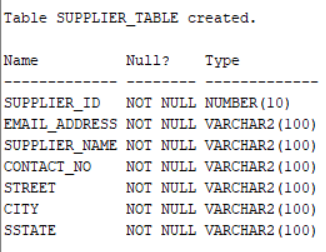
contact\_no VARCHAR2(100) NOT NULL,

street VARCHAR2(100) NOT NULL,

city VARCHAR2(100) NOT NULL,

sstate VARCHAR2(100) NOT NULL

);



**SUPPLIES\_TO\_TABLE**

CREATE TABLE SUPPLIES\_TO\_TABLE

(

publisher\_id NUMBER(10, 0) NOT NULL,

supplier\_id NUMBER(10, 0) NOT NULL,

book\_ISBN NUMBER(10, 0) NOT NULL UNIQUE,

amount NUMBER(6, 0) NOT NULL CHECK (amount >= 0),

PRIMARY KEY(publisher\_id, supplier\_id),

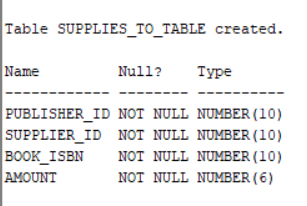
CONSTRAINT FK\_SUPPLIES\_TO\_TABLEpublisher\_id FOREIGN KEY

(publisher\_id) REFERENCES PUBLISHER\_TABLE (publisher\_id) ON DELETE CASCADE,

CONSTRAINT FK\_SUPPLIES\_TO\_TABLEsupplier\_id FOREIGN KEY

(supplier\_id) REFERENCES SUPPLIER\_TABLE (supplier\_id) ON DELETE CASCADE

);



**WAREHOUSE\_TABLE**

CREATE TABLE WAREHOUSE\_TABLE

(

warehouse\_id NUMBER(10, 0) NOT NULL PRIMARY KEY,

warehouse\_name VARCHAR2(100) NOT NULL,

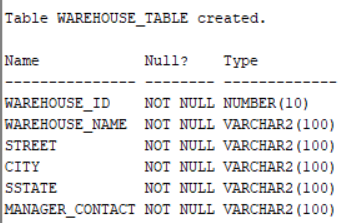
street VARCHAR2(100) NOT NULL,

city VARCHAR2(100) NOT NULL,

sstate VARCHAR2(100) NOT NULL,

manager\_contact VARCHAR2(100) NOT NULL

);



**PROVIDES\_BOOKS\_TABLE**

CREATE TABLE PROVIDES\_BOOKS\_TABLE

(

supplier\_id NUMBER(10, 0) NOT NULL,

warehouse\_id NUMBER(10, 0) NOT NULL,

book\_ISBN NUMBER(10, 0) NOT NULL UNIQUE,

amount NUMBER(6, 0) NOT NULL CHECK (amount >= 0),

PRIMARY KEY(supplier\_id, warehouse\_id),

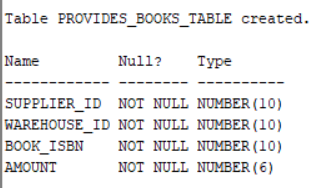
CONSTRAINT FK\_PROVIDES\_BOOKS\_TABLEsupplier\_id FOREIGN KEY

(supplier\_id) REFERENCES SUPPLIER\_TABLE (supplier\_id) ON DELETE CASCADE,

CONSTRAINT FK\_PROVIDES\_BOOKS\_TABLEwarehouse\_id FOREIGN KEY

(warehouse\_id) REFERENCES WAREHOUSE\_TABLE (warehouse\_id) ON DELETE CASCADE

);



**BOOKS\_WITH\_SUPPLIERS\_TABLE**

CREATE TABLE BOOKS\_WITH\_SUPPLIERS\_TABLE

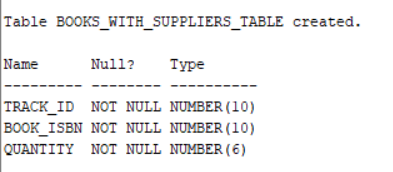
(

track\_id NUMBER(10, 0) NOT NULL PRIMARY KEY,

book\_ISBN NUMBER(10, 0) NOT NULL UNIQUE,

quantity NUMBER(6, 0) DEFAULT 0 CHECK (quantity >= 0) NOT NULL

);



**HAVE\_BOOKS\_TABLE**

CREATE TABLE HAVE\_BOOKS\_TABLE

(

supplier\_id NUMBER(10, 0) NOT NULL,

track\_id NUMBER(10, 0) NOT NULL,

PRIMARY KEY(supplier\_id, track\_id),

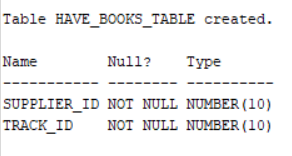
CONSTRAINT FK\_HAVE\_BOOKS\_TABLEsupplier\_id FOREIGN KEY

(supplier\_id) REFERENCES SUPPLIER\_TABLE (supplier\_id) ON DELETE CASCADE,

CONSTRAINT FK\_HAVE\_BOOKS\_TABLEtrack\_id FOREIGN KEY

(track\_id) REFERENCES BOOKS\_WITH\_SUPPLIERS\_TABLE (track\_id) ON DELETE CASCADE

);



**BOOKS\_IN\_WAREHOUSE\_TABLE**

CREATE TABLE BOOKS\_IN\_WAREHOUSE\_TABLE

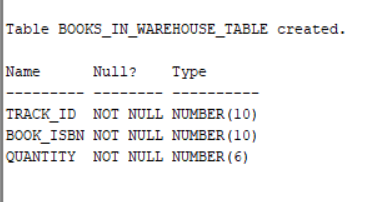
(

track\_id NUMBER(10, 0) NOT NULL PRIMARY KEY,

book\_ISBN NUMBER(10, 0) NOT NULL,

quantity NUMBER(6, 0) DEFAULT 0 CHECK (quantity >= 0) NOT NULL

);



**CONTAINS\_BOOKS\_TABLE**

CREATE TABLE CONTAINS\_BOOKS\_TABLE

(

warehouse\_id NUMBER(10, 0) NOT NULL,

track\_id NUMBER(10, 0) NOT NULL,

PRIMARY KEY(warehouse\_id, track\_id),

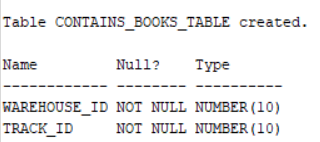
CONSTRAINT FK\_CONTAINS\_BOOKS\_TABLEwarehouse\_id FOREIGN KEY

(warehouse\_id) REFERENCES WAREHOUSE\_TABLE (warehouse\_id) ON DELETE CASCADE,

CONSTRAINT FK\_CONTAINS\_BOOKS\_TABLEbook\_ISBN FOREIGN KEY

(track\_id) REFERENCES BOOKS\_IN\_WAREHOUSE\_TABLE (track\_id) ON DELETE CASCADE

);



**USER\_INTERACTION\_TABLE**

CREATE TABLE USER\_INTERACTION\_TABLE

(

username VARCHAR2(50) NOT NULL,

book\_ISBN NUMBER(10, 0) NOT NULL,

rating NUMBER(1, 0),

upvote CHAR(1),

downvote CHAR(1),

PRIMARY KEY(username, book\_ISBN),

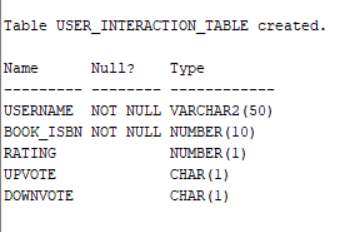
CONSTRAINT FK\_USER\_INTERACTION\_TABLEusername FOREIGN KEY

(username) REFERENCES CUSTOMER\_TABLE (username) ON DELETE CASCADE,

CONSTRAINT FK\_USER\_INTERACTION\_TABLEbook\_ISBN FOREIGN KEY

(book\_ISBN) REFERENCES BOOK\_TABLE (book\_ISBN) ON DELETE CASCADE

);



**FAVOURITES\_TABLE**

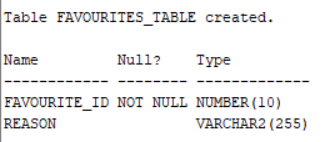
CREATE TABLE FAVOURITES\_TABLE

(

favourite\_id NUMBER(10, 0) NOT NULL PRIMARY KEY,

reason VARCHAR2(255)

);



**USER\_FAVOURITE\_TABLE**

CREATE TABLE USER\_FAVOURITE\_TABLE

(

username VARCHAR2(50) NOT NULL,

book\_ISBN NUMBER(10, 0) NOT NULL,

favourite\_id NUMBER(10, 0) NOT NULL,

PRIMARY KEY(username, book\_ISBN, favourite\_id),

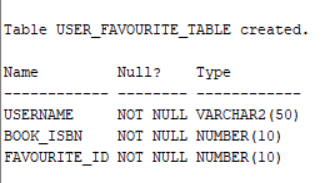
CONSTRAINT FK\_USER\_FAVOURITE\_TABLEusernamebook\_ISBN FOREIGN KEY

(username, book\_ISBN) REFERENCES USER\_INTERACTION\_TABLE (username, book\_ISBN) ON DELETE CASCADE,

CONSTRAINT FK\_USER\_FAVOURITE\_TABLEfavourite\_id FOREIGN KEY

(favourite\_id) REFERENCES FAVOURITES\_TABLE (favourite\_id) ON DELETE CASCADE

);



**COMMENTS\_TABLE**

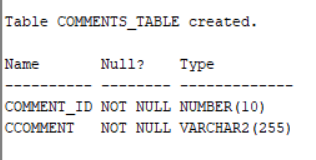
CREATE TABLE COMMENTS\_TABLE

(

comment\_id NUMBER(10, 0) NOT NULL PRIMARY KEY,

ccomment VARCHAR2(255) NOT NULL

);



**USER\_COMMENT\_TABLE**

CREATE TABLE USER\_COMMENT\_TABLE

(

username VARCHAR2(50) NOT NULL,

book\_ISBN NUMBER(10, 0) NOT NULL,

comment\_id NUMBER(10, 0) NOT NULL,

PRIMARY KEY(username, book\_ISBN, comment\_id),

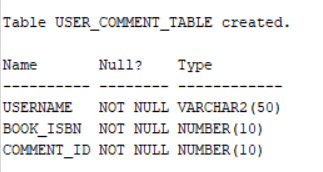
CONSTRAINT FK\_USER\_COMMENT\_TABLEusernamebook\_ISBN FOREIGN KEY

(username, book\_ISBN) REFERENCES USER\_INTERACTION\_TABLE (username, book\_ISBN) ON DELETE CASCADE,

CONSTRAINT FK\_USER\_COMMENT\_TABLEcomment\_id FOREIGN KEY

(comment\_id) REFERENCES COMMENTS\_TABLE (comment\_id) ON DELETE CASCADE

);



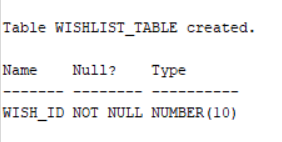
**WISHLIST\_TABLE**

CREATE TABLE WISHLIST\_TABLE

(

wish\_id NUMBER(10, 0) NOT NULL PRIMARY KEY

);



**USER\_WISHLIST\_TABLE**

CREATE TABLE USER\_WISHLIST\_TABLE

(

username VARCHAR2(50) NOT NULL,

book\_ISBN NUMBER(10, 0) NOT NULL,

wish\_id NUMBER(10, 0) NOT NULL,

PRIMARY KEY(username, book\_ISBN, wish\_id),

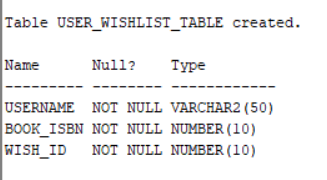
CONSTRAINT FK\_USER\_WISHLIST\_TABLEusernamebook\_ISBN FOREIGN KEY

(username, book\_ISBN) REFERENCES USER\_INTERACTION\_TABLE (username, book\_ISBN) ON DELETE CASCADE,

CONSTRAINT FK\_USER\_WISHLIST\_TABLEwish\_id FOREIGN KEY

(wish\_id) REFERENCES WISHLIST\_TABLE (wish\_id) ON DELETE CASCADE

);

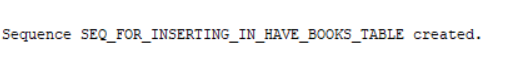


**7)Miscellaneous**

CREATE SEQUENCE SEQ\_FOR\_INSERTING\_IN\_HAVE\_BOOKS\_TABLE

START WITH 10

INCREMENT BY 10 ;



CREATE TABLE LOG\_WAREHOUSE\_TO\_SUPPLIER\_TABLE

(

log\_user VARCHAR2(50),

log\_date DATE,

from\_warehouse NUMBER(10, 0),

to\_supplier NUMBER(10, 0),

book\_sent NUMBER(10, 0),

quantity\_sent NUMBER(6, 0)

);



CREATE OR REPLACE TRIGGER TRG\_WAREHOUSE\_TO\_SUPPLIER

FOR INSERT ON PROVIDES\_BOOKS\_TABLE

COMPOUND TRIGGER

V\_track\_id NUMBER(10, 0) := &V\_track\_id ;

V\_supplier\_id NUMBER(10, 0) := &V\_supplier\_id ;

AFTER EACH ROW IS

BEGIN

IF (:NEW.amount >= 1000) THEN

RAISE\_APPLICATION\_ERROR(-20005,'You cannot provide more than 1000 books...');

END IF ;

INSERT INTO BOOKS\_WITH\_SUPPLIERS\_TABLE VALUES(SEQ\_FOR\_INSERTING\_IN\_HAVE\_BOOKS\_TABLE.NEXTVAL, :NEW.book\_ISBN, :NEW.amount);

INSERT INTO HAVE\_BOOKS\_TABLE VALUES(V\_supplier\_id, SEQ\_FOR\_INSERTING\_IN\_HAVE\_BOOKS\_TABLE.CURRVAL);

UPDATE BOOKS\_IN\_WAREHOUSE\_TABLE

SET quantity = quantity - :NEW.amount

WHERE track\_id = V\_track\_id AND book\_ISBN = :NEW.book\_ISBN ;

IF (BOOKS\_IN\_WAREHOUSE\_TABLE.quantity = 0) THEN

DELETE FROM BOOKS\_IN\_WAREHOUSE\_TABLE WHERE quantity = 0 ;

END IF ;

INSERT INTO LOG\_WAREHOUSE\_TO\_SUPPLIER\_TABLE VALUES(USER, SYSDATE, :NEW.warehouse\_id, V\_supplier\_id, :NEW.amount);

END AFTER EACH ROW ;

AFTER STATEMENT IS

BEGIN

DELETE FROM PROVIDES\_BOOKS\_TABLE ;

DBMS\_OUTPUT.PUT\_LINE('Required work has been done successfully!');

END AFTER STATEMENT ;

END ;

CREATE TABLE LOG\_SUPPLIER\_TO\_PUBLISHER\_TABLE

(

log\_user VARCHAR2(50),

log\_date DATE,

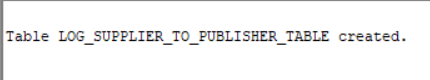
from\_supplier NUMBER(10, 0),

to\_publisher NUMBER(10, 0),

book\_sent NUMBER(10, 0),

quantity\_sent NUMBER(6, 0)

);



CREATE OR REPLACE FUNCTION FUN\_BOOK\_EXISTS (search\_book\_ISBN BOOK\_TABLE.book\_ISBN%type) RETURN BOOLEAN AS

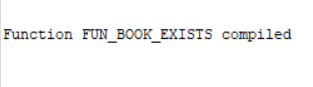
v\_count NUMBER ;

BEGIN

SELECT COUNT(\*) INTO v\_count FROM BOOK\_TABLE WHERE book\_ISBN = search\_book\_ISBN ;

RETURN (v\_count > 0) ;

END FUN\_BOOK\_EXISTS ;



CREATE OR REPLACE TRIGGER TRG\_SUPPLIER\_TO\_PUBLISHER

FOR INSERT ON SUPPLIES\_TO\_TABLE

COMPOUND TRIGGER

v\_book\_title VARCHAR2(255) ;

v\_edition NUMBER(2, 0) ;

v\_price NUMBER(7, 3) ;

v\_eoq NUMBER(7, 3) ;

v\_author\_id NUMBER(10, 0) ;

v\_release\_date DATE ;

v\_page\_count NUMBER(4, 0) ;

v\_dimensions VARCHAR2(50) ;

BEFORE EACH ROW IS

BEGIN

IF FUN\_BOOK\_EXISTS(:NEW.book\_ISBN) THEN

UPDATE BOOK\_TABLE

SET quantity\_in\_stock = quantity\_in\_stock + :NEW.amount

WHERE book\_ISBN = :NEW.book\_ISBN ;

ELSE

v\_book\_title := &v\_book\_title ;

v\_edition := &v\_edition ;

v\_price := &v\_price ;

v\_eoq := &v\_eoq ;

v\_author\_id := &v\_author\_id ;

v\_release\_date := &v\_release\_date ;

v\_page\_count := &v\_page\_count ;

v\_dimensions := &v\_dimensions ;

INSERT INTO BOOK\_TABLE VALUES(:NEW.book\_ISBN, v\_book\_title, v\_edition, v\_price,

v\_eoq, v\_author\_id, :NEW.publisher\_id, v\_release\_date, v\_page\_count, v\_dimensions, :NEW.amount) ;

END IF ;

INSERT INTO LOG\_SUPPLIER\_TO\_PUBLISHER\_TABLE VALUES(USER, SYSDATE, :NEW.supplier\_id, :NEW.publisher\_id, :NEW.book\_ISBN, :NEW.amount) ;

END BEFORE EACH ROW ;

AFTER STATEMENT IS

BEGIN

DELETE FROM SUPPLIES\_TO\_TABLE ;

DBMS\_OUTPUT.PUT\_LINE('Required work has been done successfully!');

END AFTER STATEMENT ;

END ;