NITW

Computer Science Engineering

ONLINE BOOKSTORE MANAGEMENT SYSTEM PROJECT

KOMAL KUMARI (21CSB0B28)

KONDRU YUV RAJ (21CSBOB29)

CSE 2ND YEAR (2022-23)

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.

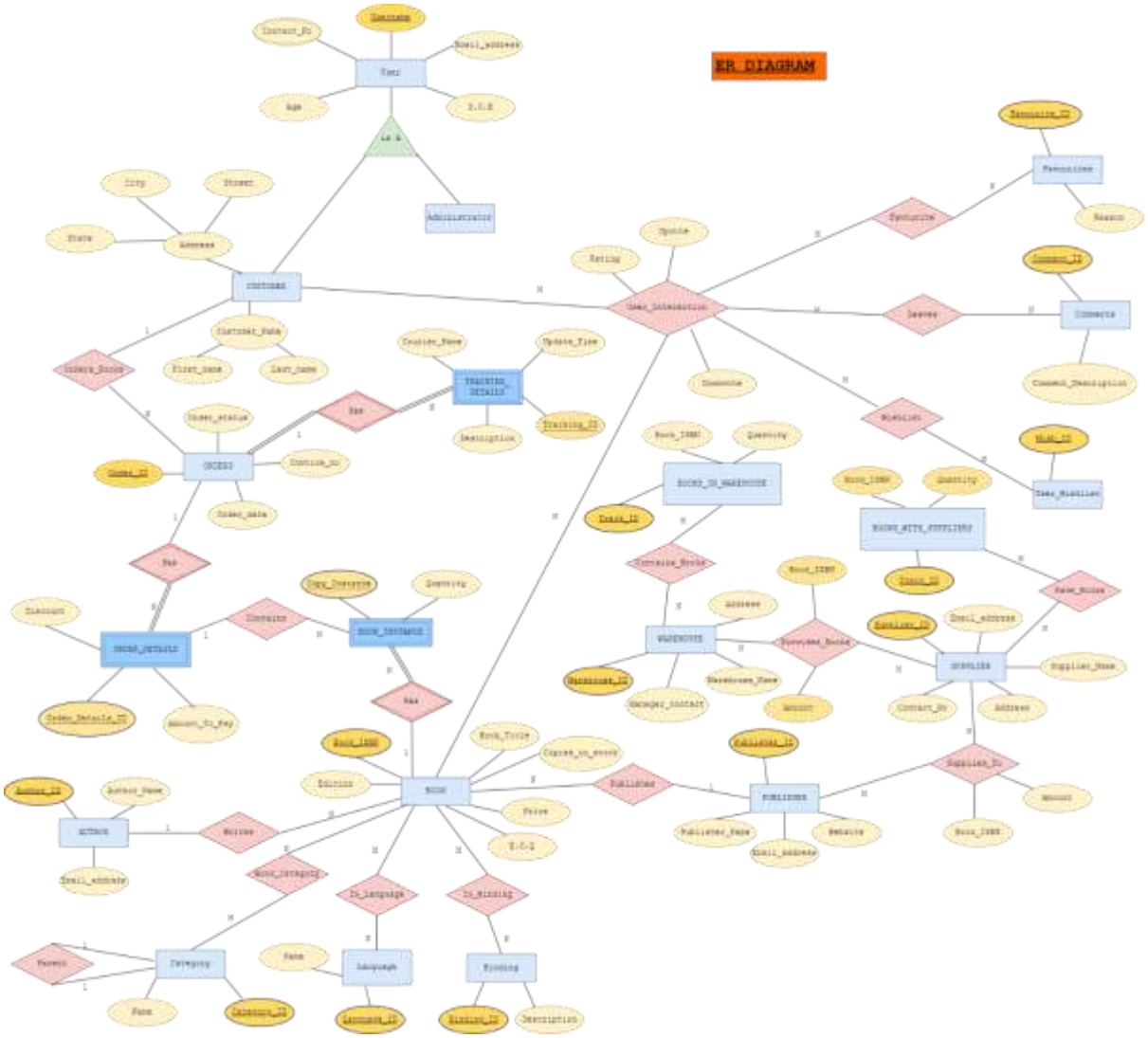
The database should be scalable to handle large number of users and a growing amount of data. The database should be designed to ensure data reliability and database consistency. The final solution should be presented in the form of an ER model and a database implementation using a RDBMS such as SQL.

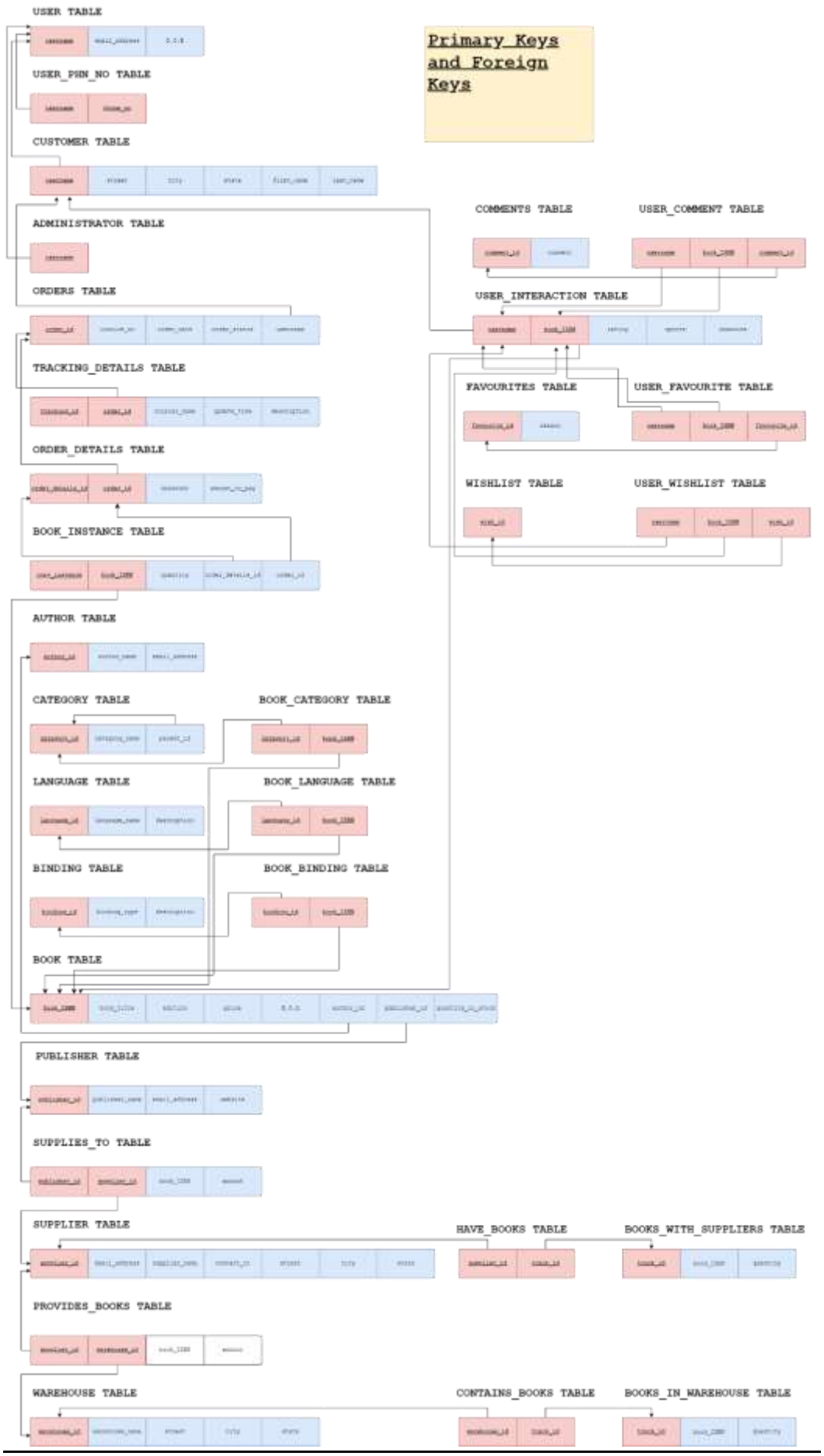
CONTENTS:

1) ER Model Assumptions

- 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

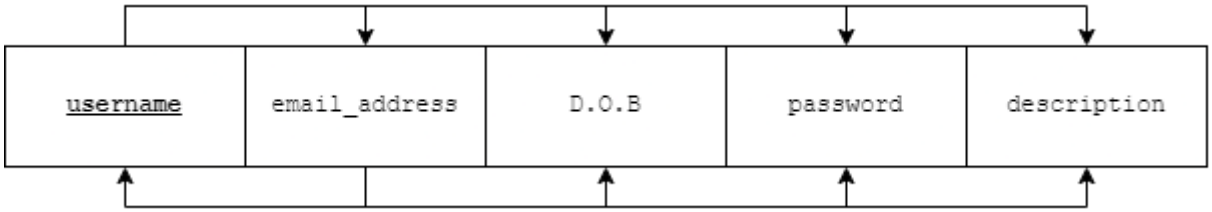



Primary Keys and Foreign Keys
USER TABLE
USER_PHN_NO TABLE
CUSTOMER TABLE
ADMINISTRATOR TABLE
ORDERS TABLE
TRACKING_DETAILS TABLE
ORDER_DETAILS TABLE
BOOK_INSTANCE TABLE
AUTHOR TABLE
CATEGORY TABLE
BOOK_CATEGORY TABLE
LANGUAGE TABLE
BOOK_LANGUAGE TABLE
BINDING TABLE
BOOK_BINDING TABLE
BOOK TABLE
PUBLISHER TABLE
SUPPLIES_TO TABLE
SUPPLIER TABLE
HAVE_BOOKS TABLE
BOOKS_WITH_SUPPLIERS TABLE
PROVIDES_BOOKS TABLE
WAREHOUSE TABLE
CONTAINS_BOOKS TABLE
BOOKS_IN_WAREHOUSE TABLE
COMMENTS TABLE
USER_COMMENT TABLE
USER_INTERACTION TABLE
FAVOURITES TABLE
USER_FAVOURITE TABLE
WISHLIST TABLE
USER_WISHLIST TABLE

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.

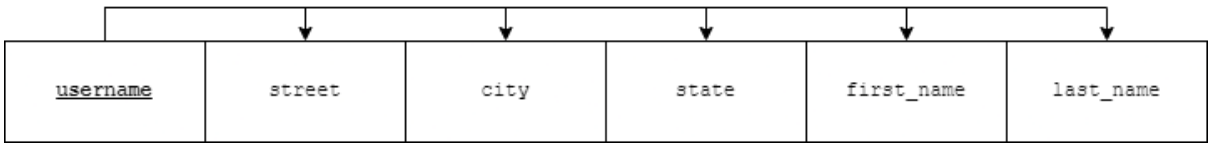
USER_PHN_NO TABLE

username	phone_no

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}

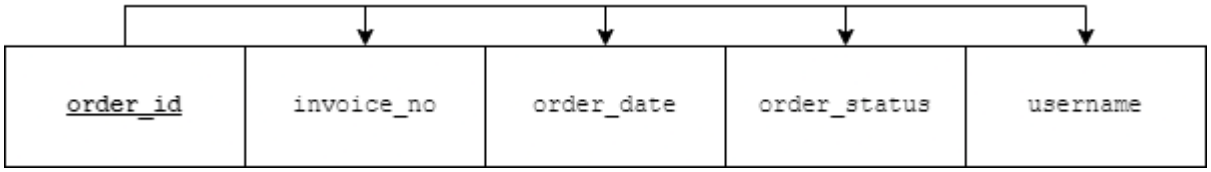
There is only one functional dependency.
It satisfies 1NF as all attributes are atomic.
It also satisfies 2NF as there are no partial dependencies, no non prime attribute is dependent on any proper subset of the candidate keys. (Candidate key is atomic)
It also satisfies 3NF as there are no transitive 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}



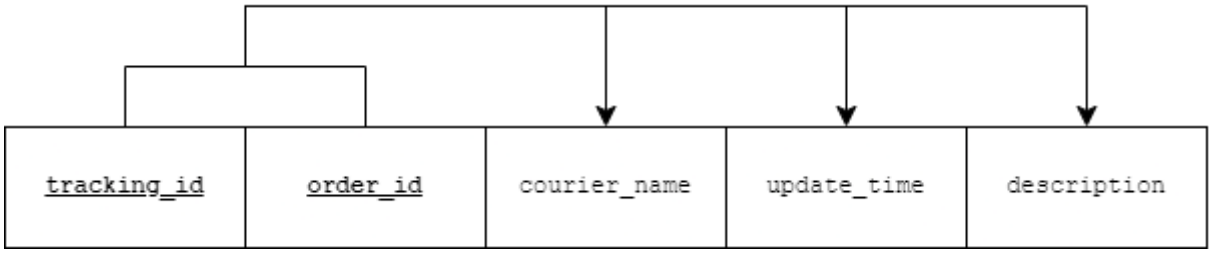
order_id is a candidate key.

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}



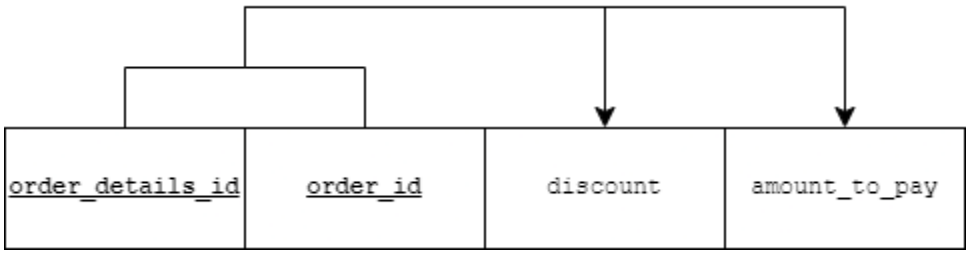
tracking_id order_id is the candidate key.

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}



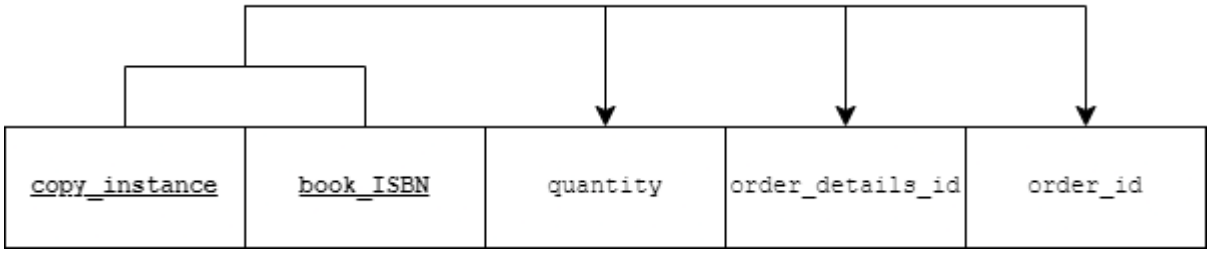
order_details_id order_id is the candidate key.

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}



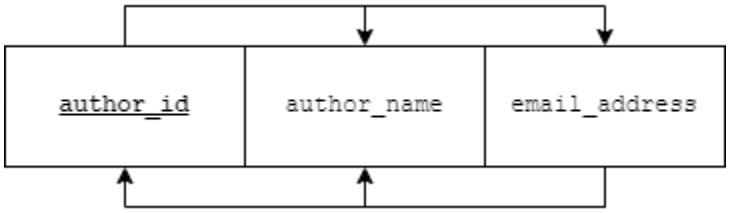
copy_instance book_ISBN is the candidate key.

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.

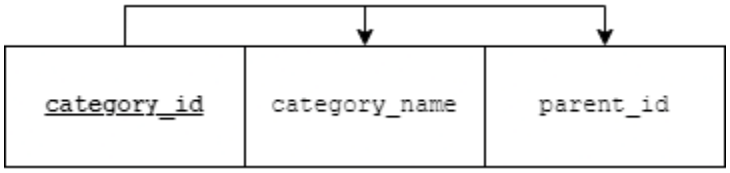
Prime attributes = {author_id, email_address}
Non prime attributes = {author_name}

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}

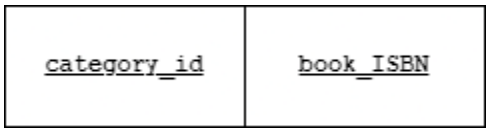


category_id is a candidate key.

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.

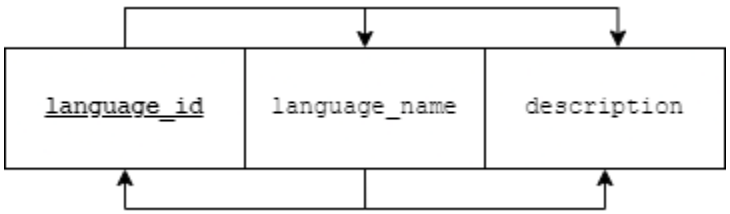
BOOK_CATEGORY TABLE



There are only two prime attributes category_id and book_ISBN.
It is automatically in BCNF because all dependencies are trivial.

LANGUAGE TABLE

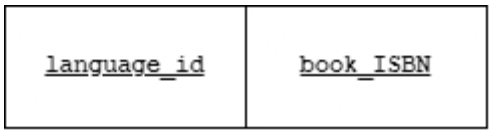
language_id = {language_name, description}
language_name = {language_id, description}



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.

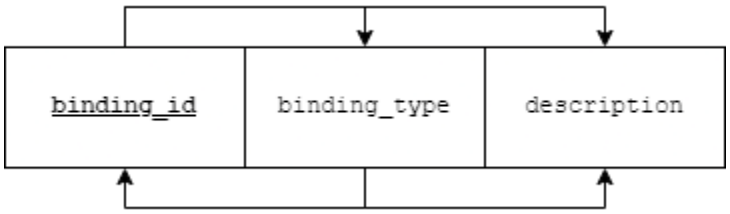
BOOK_LANGUAGE TABLE



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

binding_id	book_ISBN

There are only two prime attributes binding_id and book_ISBN.
It is automatically in BCNF because all dependencies are trivial.

BOOK TABLE

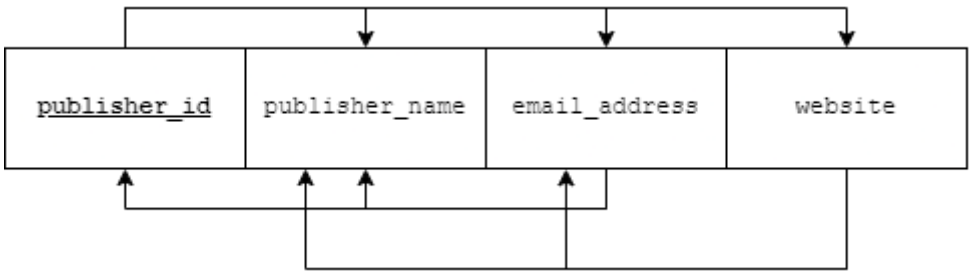
book_ISBN -> {book_title, edition, price, E.O.Q, author_id, publisher_id, release_date, page_count, dimension, quantity_in_stock}

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}

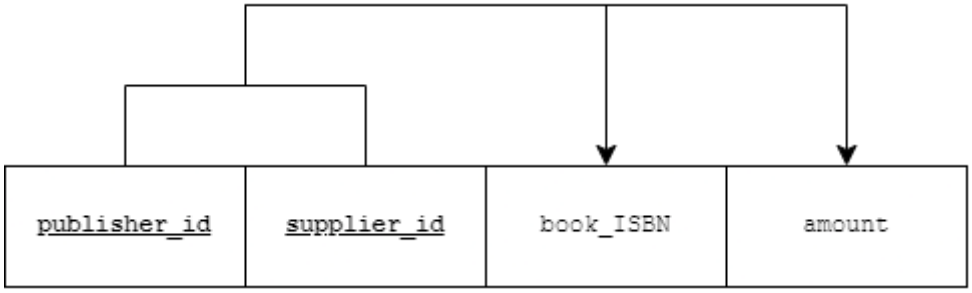


If we check their closures, publisher_id, email_address, website are candidate keys.

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}



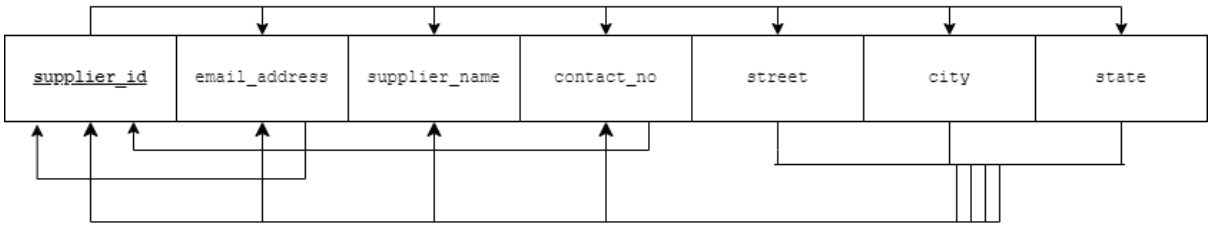
publisher_id supplier_id is the candidate key.
Prime attributes = {publisher_id, supplier_id}
Non prime attributes = {book_ISBN, amount}

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}



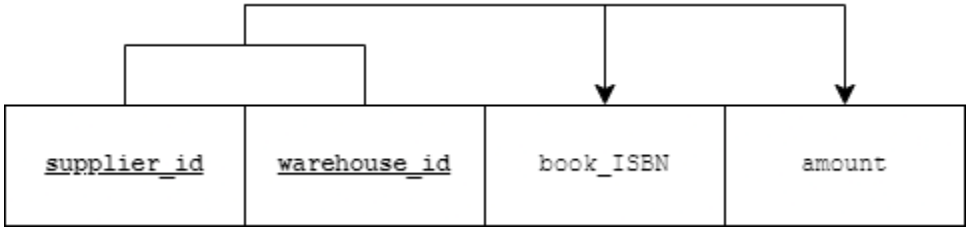
supplier_id, contact_no, email_address and street, city, state are candidate keys.

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}

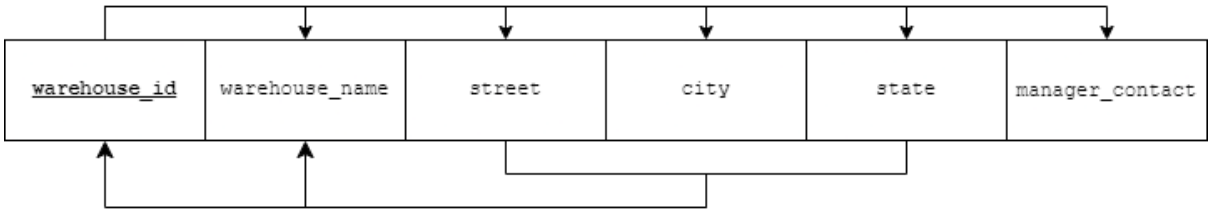


supplier_id warehouse_id is the candidate key.
Prime attributes = {supplier_id, warehouse_id}
Non prime attributes = {book_ISBN, amount}

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}

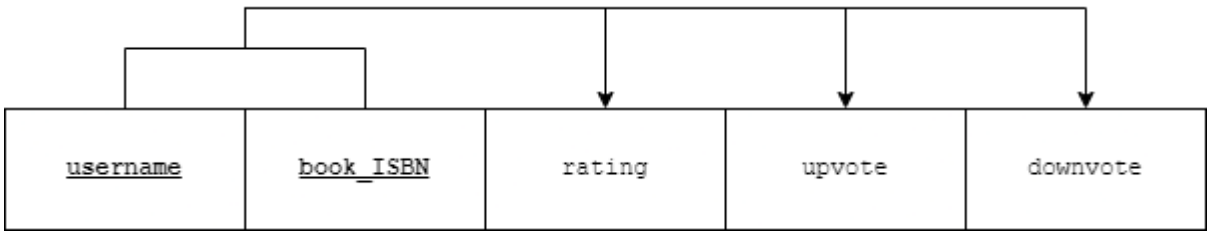


warehouse_id, street city state are the candidate keys.
Prime attributes = {warehouse_id, street, city, state}
Non prime attributes = {warehouse_name}

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}

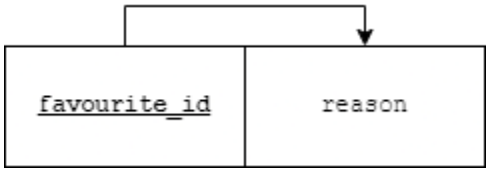


username book_ISBN is the candidate key.
Prime attributes = {username, book_ISBN}
Non prime attributes = {rating, upvote, downvote}

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.

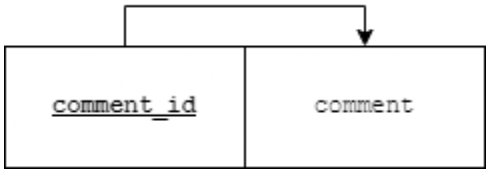
USER_FAVOURITE TABLE

username	book_ISBN	favourite_id

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

username	book_ISBN	comment_id

All are prime attributes.

It is automatically in BCNF as all dependencies are trivial.

WISHLIST_TABLE

wish_id

There is only one prime attribute.
It is automatically in BCNF as all dependencies are trivial.

USER_WISHLIST TABLE

username	book_ISBN	wish_id

All are prime attributes.
It is automatically in BCNF as all dependencies are trivial.

HAVE_BOOKS TABLE

supplier_id	track_id

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	book_ISBN	quantity

track_id is the candidate key.

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

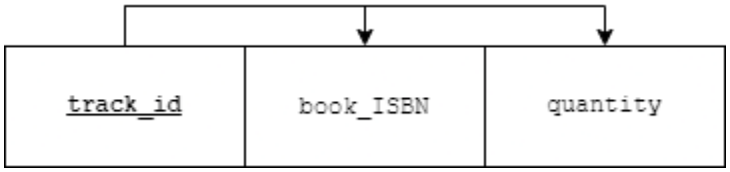
CONTAINS_BOOKS TABLE

warehouse_id	track_id

All are prime attributes.
It is automatically in BCNF as all dependencies are trivial.

BOOKS_IN_WAREHOUSE TABLE

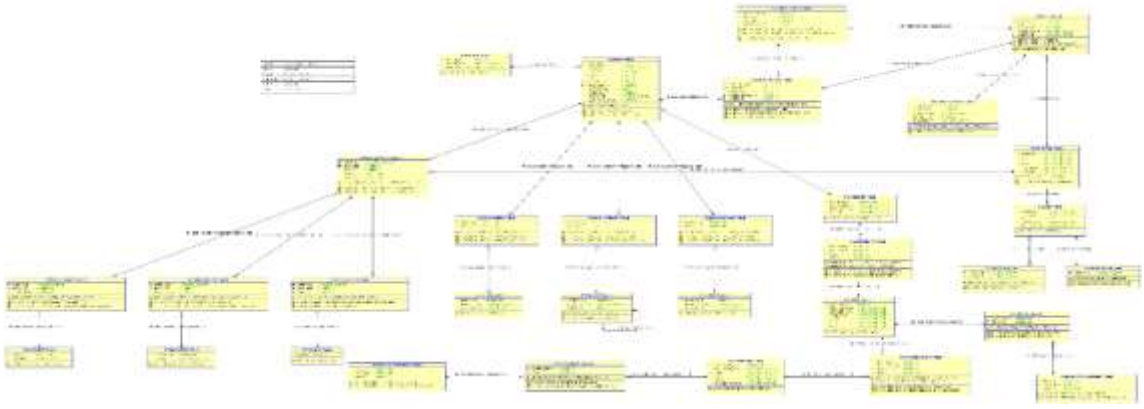
track_id = {book_ISBN, quantity}



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.

2. 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.

3. 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.

4. Data Dictionary for ADMINISTRATOR

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

5. 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	

6. 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.

7. 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

8. 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	

9. Data Dictionary for AUTHOR

Column Status	Attribute	Data type	Size	Description
Not Null	author_id	NUMBER	10, 0	Unique ID for each author.

Primary Key				
Not Null	author_name	VARCHAR2	50	Name of the author.
Not Null	email_address	VARCHAR2	100	Email address of the author

10. 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.

11. 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.

12. 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	

13. 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	

14. 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	

15. 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	

16. 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	

17. 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	

18. 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	

19. 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	

20. 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	warehouse_id	NUMBER	10, 0	

Composite Key Foreign Key				
Not Null Unique	book_ISBN	NUMBER	10, 0	
Not Null	amount	NUMBER	6, 0	Amount of books.

21. 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	

22. 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	

23. Data Dictionary for BOOKS_WITH_SUPPLIERS

Column Status	Attribute	Data type	Size	Description
Not Null Primary Key	track_id	NUMBER	10, 0	
Not Null	book_ISBN	NUMBER	10, 0	

Unique				
Not Null	quantity	NUMBER	6, 0	

24. 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	

25. 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	

26. Data Dictionary for USER_INTERACTION

Column Status	Attribute	Data type	Size	Description
Not Null Composite Key Foreign Key	username	VARCHAR2	50	
Not Null	book_ISBN	NUMBER	10, 0	

Composite Key Foreign Key				
	rating	NUMBER	1, 0	
	upvote	CHAR	1	
	downvote	CHAR	1	

27. Data Dictionary for FAVOURITES

Column Status	Attribute	Data type	Size	Description
Not Null Primary Key	favourite_id	NUMBER	10, 0	
	reason	VARCHAR2	255	

28. 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	

29. 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	

30. 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	

31. Data Dictionary for WISHLIST

Column Status	Attribute	Data type	Size	Description
Not Null Primary Key	wish_id	NUMBER	10, 0	

32. 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)
);
```

```
Table USER_TABLE created.

Name           Null?    Type
-------------- -------- -------------
USERNAME       NOT NULL VARCHAR2(50)
EMAIL_ADDRESS           VARCHAR2(100)
DOB                     DATE
PPASSWORD      NOT NULL VARCHAR2(20)
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
);
```

```
Table USER_PHN_NO_TABLE created.

Name     Null?    Type
-------- -------- -------------
USERNAME NOT NULL VARCHAR2(50)
PHONE_NO NOT NULL VARCHAR2(100)
```

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
);
```

```
Table CUSTOMER_TABLE created.

Name       Null?    Type
---------- -------- -------------
USERNAME   NOT NULL VARCHAR2(50)
STREET     NOT NULL VARCHAR2(100)
CITY       NOT NULL VARCHAR2(100)
SSTATE     NOT NULL VARCHAR2(100)
FIRST_NAME          VARCHAR2(50)
LAST_NAME           VARCHAR2(50)
```

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
);
```

```
Table ADMINISTRATOR_TABLE created.

Name     Null?    Type
-------- -------- ------------
USERNAME NOT NULL VARCHAR2(50)
```

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
);
```

```
Table ORDERS_TABLE created.

Name         Null?    Type
------------ -------- -------------
ORDER_ID     NOT NULL NUMBER(10)
INVOICE_NO   NOT NULL NUMBER(10)
ORDER_DATE   NOT NULL TIMESTAMP(6)
ORDER_STATUS NOT NULL VARCHAR2(100)
USERNAME     NOT NULL VARCHAR2(50)
```

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
);
```

```
Table TRACKING_DETAILS_TABLE created.

Name         Null?    Type
------------ -------- -------------
TRACKING_ID  NOT NULL NUMBER(10)
ORDER_ID     NOT NULL NUMBER(10)
COURIER_NAME NOT NULL VARCHAR2(100)
UPDATE_TIME  NOT NULL TIMESTAMP(6)
DDESCRIPTION NOT NULL VARCHAR2(100)
```

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
);
```

```
Table ORDER_DETAILS_TABLE created.

Name             Null?    Type
---------------- -------- ------------
ORDER_DETAILS_ID NOT NULL NUMBER(10)
ORDER_ID         NOT NULL NUMBER(10)
DISCOUNT         NOT NULL NUMBER(5,2)
AMOUNT_TO_PAY    NOT NULL NUMBER(10,3)
```

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
);
```

```
Table AUTHOR_TABLE created.

Name          Null?    Type
------------- -------- -------------
AUTHOR_ID     NOT NULL NUMBER(10)
AUTHOR_NAME   NOT NULL VARCHAR2(50)
EMAIL_ADDRESS NOT NULL VARCHAR2(100)
```

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
);
```

```
Table PUBLISHER_TABLE created.

Name           Null?    Type
-------------- -------- -------------
PUBLISHER_ID   NOT NULL NUMBER(10)
PUBLISHER_NAME NOT NULL VARCHAR2(100)
EMAIL_ADDRESS  NOT NULL VARCHAR2(100)
WEBSITE        NOT NULL VARCHAR2(100)
```

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
);
```

```
Table BOOK_TABLE created.

Name                 Null?    Type
-------------------- -------- -------------
BOOK_ISBN            NOT NULL NUMBER(10)
BOOK_TITLE                    VARCHAR2(255)
EEDITION                      NUMBER(2)
PRICE                NOT NULL NUMBER(7,3)
EOQ                           NUMBER(7,3)
AUTHOR_ID            NOT NULL NUMBER(10)
PUBLISHER_ID         NOT NULL NUMBER(10)
RELEASE_DATE                  DATE
PAGE_COUNT                    NUMBER(4)
DIMENSIONS                    VARCHAR2(50)
QUANTITY_IN_STOCK NOT NULL NUMBER(6)
```

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
);
```

```
Table BOOK_INSTANCE_TABLE created.

Name                 Null?    Type
---------------- -------- ----------
COPY_INSTANCE        NOT NULL NUMBER(10)
BOOK_ISBN            NOT NULL NUMBER(10)
QUANTITY             NOT NULL NUMBER(5)
ORDER_DETAILS_ID NOT NULL NUMBER(10)
ORDER_ID             NOT NULL NUMBER(10)
```

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
);
```

```
Table CATEGORY_TABLE created.

Name          Null?    Type
------------- -------- ------------
CATEGORY_ID   NOT NULL NUMBER(10)
CATEGORY_NAME NOT NULL VARCHAR2(50)
PARENT_ID              NUMBER(10)
```

LANGUAGE_TABLE

```
CREATE TABLE LANGUAGE_TABLE
(
   language_id NUMBER(10, 0) NOT NULL PRIMARY KEY,
   language_name VARCHAR2(50) NOT NULL,
   ddescription VARCHAR2(255)
);
```

```
Table LANGUAGE_TABLE created.

Name          Null?    Type
------------- -------- -------------
LANGUAGE_ID   NOT NULL NUMBER(10)
LANGUAGE_NAME NOT NULL VARCHAR2(50)
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)
);
```

```
Table BINDING_TABLE created.

Name         Null?    Type
------------ -------- -------------
BINDING_ID   NOT NULL NUMBER(10)
BINDING_TYPE NOT NULL VARCHAR2(50)
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
);
```

```
Table BOOK_CATEGORY_TABLE created.

Name        Null?    Type
----------- -------- ----------
CATEGORY_ID NOT NULL NUMBER(10)
BOOK_ISBN   NOT NULL NUMBER(10)
```

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
);
```

```
Table BOOK_LANGUAGE_TABLE created.

Name        Null?    Type
----------- -------- ----------
LANGUAGE_ID NOT NULL NUMBER(10)
BOOK_ISBN   NOT NULL NUMBER(10)
```

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
);
```

```
Table BOOK_BINDING_TABLE created.

Name       Null?    Type
---------- -------- ----------
BINDING_ID NOT NULL NUMBER(10)
BOOK_ISBN  NOT NULL NUMBER(10)
```

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
);
```

```
Table SUPPLIER_TABLE created.

Name          Null?    Type
------------- -------- -------------
SUPPLIER_ID   NOT NULL NUMBER(10)
EMAIL_ADDRESS NOT NULL VARCHAR2(100)
SUPPLIER_NAME NOT NULL VARCHAR2(100)
CONTACT_NO    NOT NULL VARCHAR2(100)
STREET        NOT NULL VARCHAR2(100)
CITY          NOT NULL VARCHAR2(100)
SSTATE        NOT NULL VARCHAR2(100)
```

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
);
```

```
Table SUPPLIES_TO_TABLE created.

Name         Null?    Type
------------ -------- ----------
PUBLISHER_ID NOT NULL NUMBER(10)
SUPPLIER_ID  NOT NULL NUMBER(10)
BOOK_ISBN    NOT NULL NUMBER(10)
AMOUNT       NOT NULL NUMBER(6)
```

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
);
```

```
Table WAREHOUSE_TABLE created.

Name            Null?    Type
--------------- -------- -------------
WAREHOUSE_ID    NOT NULL NUMBER(10)
WAREHOUSE_NAME  NOT NULL VARCHAR2(100)
STREET          NOT NULL VARCHAR2(100)
CITY            NOT NULL VARCHAR2(100)
SSTATE          NOT NULL VARCHAR2(100)
MANAGER_CONTACT NOT NULL VARCHAR2(100)
```

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
);
```

```
Table PROVIDES_BOOKS_TABLE created.

Name         Null?    Type
------------ -------- ----------
SUPPLIER_ID  NOT NULL NUMBER(10)
WAREHOUSE_ID NOT NULL NUMBER(10)
BOOK_ISBN    NOT NULL NUMBER(10)
AMOUNT       NOT NULL NUMBER(6)
```

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
);
```

```
Table BOOKS_WITH_SUPPLIERS_TABLE created.

Name      Null?    Type
--------- -------- ----------
TRACK_ID  NOT NULL NUMBER(10)
BOOK_ISBN NOT NULL NUMBER(10)
QUANTITY  NOT NULL NUMBER(6)
```

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
);
```

```
Table HAVE_BOOKS_TABLE created.

Name        Null?    Type
----------- -------- ----------
SUPPLIER_ID NOT NULL NUMBER(10)
TRACK_ID    NOT NULL NUMBER(10)
```

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
);
```

```
Table BOOKS_IN_WAREHOUSE_TABLE created.

Name      Null?    Type
--------- -------- ----------
TRACK_ID  NOT NULL NUMBER(10)
BOOK_ISBN NOT NULL NUMBER(10)
QUANTITY  NOT NULL NUMBER(6)
```

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
);
```

```
Table CONTAINS_BOOKS_TABLE created.

Name         Null?    Type
------------ -------- ----------
WAREHOUSE_ID NOT NULL NUMBER(10)
TRACK_ID     NOT NULL NUMBER(10)
```

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
);
```

```
Table USER_INTERACTION_TABLE created.

Name      Null?    Type
--------- -------- ------------
USERNAME  NOT NULL VARCHAR2(50)
BOOK_ISBN NOT NULL NUMBER(10)
RATING             NUMBER(1)
UPVOTE             CHAR(1)
DOWNVOTE           CHAR(1)
```

FAVOURITES_TABLE

```
CREATE TABLE FAVOURITES_TABLE
(
   favourite_id NUMBER(10, 0) NOT NULL PRIMARY KEY,
   reason VARCHAR2(255)
);
```

```
Table FAVOURITES_TABLE created.

Name         Null?    Type
------------ -------- -------------
FAVOURITE_ID NOT NULL NUMBER(10)
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
);
```

```
Table USER_FAVOURITE_TABLE created.

Name          Null?    Type
------------- -------- ------------
USERNAME      NOT NULL VARCHAR2(50)
BOOK_ISBN     NOT NULL NUMBER(10)
FAVOURITE_ID NOT NULL NUMBER(10)
```

COMMENTS_TABLE

```
CREATE TABLE COMMENTS_TABLE
(
   comment_id NUMBER(10, 0) NOT NULL PRIMARY KEY,
   ccomment VARCHAR2(255) NOT NULL
);
```

```
Table COMMENTS_TABLE created.

Name        Null?    Type
----------- -------- -------------
COMMENT_ID NOT NULL NUMBER(10)
CCOMMENT   NOT NULL VARCHAR2(255)
```

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
);
```

```
Table USER_COMMENT_TABLE created.

Name        Null?    Type
----------- -------- ------------
USERNAME    NOT NULL VARCHAR2(50)
BOOK_ISBN   NOT NULL NUMBER(10)
COMMENT_ID  NOT NULL NUMBER(10)
```

WISHLIST_TABLE

```
CREATE TABLE WISHLIST_TABLE
(
   wish_id NUMBER(10, 0) NOT NULL PRIMARY KEY
);
```

```
Table WISHLIST_TABLE created.

Name    Null?    Type
------- -------- ----------
WISH_ID NOT NULL NUMBER(10)
```

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
);
```

```
Table USER_WISHLIST_TABLE created.

Name      Null?    Type
--------- -------- ------------
USERNAME  NOT NULL VARCHAR2(50)
BOOK_ISBN NOT NULL NUMBER(10)
WISH_ID   NOT NULL NUMBER(10)
```

7)Miscellaneous

```
CREATE SEQUENCE SEQ_FOR_INSERTING_IN_HAVE_BOOKS_TABLE
START WITH 10
INCREMENT BY 10 ;
```

```
Sequence SEQ_FOR_INSERTING_IN_HAVE_BOOKS_TABLE created.
```

```
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)
);
```

```
Table LOG_WAREHOUSE_TO_SUPPLIER_TABLE created.
```

```
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)
);
```

```
Table LOG_SUPPLIER_TO_PUBLISHER_TABLE created.
```

```
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 ;
```

```
Function FUN_BOOK_EXISTS compiled
```

```
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 ;
```