Assignment 1: Design a database schema for a library system, including tables,fields, and constraints like NOT NULL, UNIQUE, and CHECK. Include primary and foreign keys to establish relationships between tables.

->

CREATE TABLE Authors (

author\_id NUMBER PRIMARY KEY,

author\_name VARCHAR2(100) UNIQUE NOT NULL,

birth\_date DATE,

nationality VARCHAR2(50)

);

CREATE TABLE Books (

book\_id NUMBER PRIMARY KEY,

title VARCHAR2(200) NOT NULL,

author\_id NUMBER,

isbn VARCHAR2(20) UNIQUE CHECK (isbn LIKE 'ISBN-%'),

publication\_date DATE,

FOREIGN KEY (author\_id) REFERENCES Authors(author\_id)

);

CREATE TABLE Genres (

genre\_id NUMBER PRIMARY KEY,

genre\_name VARCHAR2(100) UNIQUE NOT NULL

);

CREATE TABLE Book\_Genres (

book\_id NUMBER,

genre\_id NUMBER,

PRIMARY KEY (book\_id, genre\_id),

FOREIGN KEY (book\_id) REFERENCES Books(book\_id),

FOREIGN KEY (genre\_id) REFERENCES Genres(genre\_id)

);

CREATE TABLE Borrowers (

borrower\_id NUMBER PRIMARY KEY,

borrower\_name VARCHAR2(100) NOT NULL,

contact\_number VARCHAR2(20),

email VARCHAR2(100) UNIQUE NOT NULL

);

CREATE TABLE Transactions (

transaction\_id NUMBER PRIMARY KEY,

book\_id NUMBER,

borrower\_id NUMBER,

issue\_date DATE NOT NULL,

due\_date DATE NOT NULL,

return\_date DATE,

fine\_amount NUMBER CHECK (fine\_amount >= 0),

FOREIGN KEY (book\_id) REFERENCES Books(book\_id),

FOREIGN KEY (borrower\_id) REFERENCES Borrowers(borrower\_id)

);

Assignment 2:

1.Find the product whose category is electronics.

-> SELECT \*

FROM Products

WHERE category\_id IN (SELECT category\_id FROM Categories WHERE category\_name = 'electronic');

2.Find all products whose price is between 5000 and 10000

-> SELECT \*

FROM Products

WHERE price BETWEEN 5000 AND 10000;

3.Display name and price from product table

-> SELECT name, price

FROM Products;

4.Display all electronic category items

-> SELECT \*

FROM Products

WHERE category\_id = (SELECT category\_id FROM Categories WHERE category\_name = 'electronic');

5.Sort the products based on price in descending order

-> SELECT \*

FROM Products

ORDER BY price DESC;

6.Sort the products based on name

-> SELECT \*

FROM Products

ORDER BY name;

7.Count of products based on category

-> SELECT c.category\_name, COUNT(p.product\_id) AS product\_count

FROM Categories c

LEFT JOIN Products p ON c.category\_id = p.category\_id

GROUP BY c.category\_name;

8.Display all products except electronics category

-> SELECT \*

FROM Products

WHERE category\_id != (SELECT category\_id FROM Categories WHERE category\_name = 'electronics');

Assignment 3:

1. Select the names of all the products in the store.

-> Select name from product;

1. Select the names and the prices of all the products in the store.

-> Select name, price from product;

1. Select the name of the products with a price less than or equal to $200.

-> Select name from product where price <= $200;

1. Select all the products with a price between $60 and $120.

-> select \*from product where price between $60 and $120;

1. Select the name and price in cents (i.e., the price must be multiplied by 100).

-> select name, price\*100 from products;

-> select name, concat(price\*100, ' cents') from products;

1. Compute the average price of all the products.

-> select avg(price) from products;

-> select sum(price)/count(price) from products;

1. Compute the average price of all products with manufacturer code equal to 2.

-> select avg(price) from product where manufacturer =2

1. Compute the number of products with a price larger than or equal to $180.

-> select count(\*) product where price >= 180

1. Select the name and price of all products with a price larger than or equal to $180, and sort first by price (in descending order), and then by name (in ascending order).

-> select name, price from product where price >= 180 order by price desc, name asc;

1. Select all the data from the products, including all the data for each product's manufacturer.

-> select a.\*, b.name from products a join Manufacturers b on(a.manufacturer = b.code);

-> select a.\*, b.name from products a, Manufacturers b where a.manufacturer = b.code;

1. Select the product name, price, and manufacturer name of all the products.

-> select a.name, a.price, b.name from products a join Manufacturers b on(a.manufacturer = b.code);

-> select products.name, Price, Manufacturers.name

From products inner join Manufacturers

on products.manufacturer = manufacturers.code;

1. Select the average price of each manufacturer's products, showing only the manufacturer's code.

-> select avg(price), manufacturer

From products

Group by manufacturer;

1. Select the average price of each manufacturer's products, showing the manufacturer's name.

-> select avg(a.price), b.name

from Products a join Manufacturers b

on a.manufacturer = b.code

group by b.name;

1. Select the names of manufacturer whose products have an average price larger than or equal to $150.

-> select avg(a.price), b.name

from manufacturers b join Products a

on b.code = a.manufacturer

group by b.name

having avg(a.price)>=150;

-> select avg(price), manufacturers.name

From products, manufacturers

where products.manufacturer = manufacturers.code

group by manufacturers.name

having avg (price) >= 150;

1. Select the name and price of the cheapest product.

-> select name, price from Products

where price = (

select min(price)

from products);

-> select name,price

from products

order by price ASC

limit 1;

select top 1 name, price

from products

order by price ASC

1. Select the name of each manufacturer along with the name and price of its most expensive product.

-> select max\_price\_mapping.name as manu\_name, max\_price\_mapping.price, products\_with\_manu\_name.name as product\_name

from

(select manufacturers.name, max(Price) price

from products, manufacturers

where manufacturer = manufacturers.code

group by manufacturers.name)

as max\_price\_mapping

left join

(select products.\*, manufacturers.name manu\_name

from products join manufacturers

on (products.manufacturer = manufacturers.code))

as products\_with\_manu\_name

on

(max\_price\_mapping.name = products\_with\_manu\_name.manu\_name

and

max\_price\_mapping.price = products\_with\_manu\_name.price);

1. Add a new product: Loudspeakers, $70, manufacturer 2.

-> insert into Products values (11, 'Loudspeakers', 70, 2);

1. Update the name of product 8 to "Laser Printer".

-> update products

set name = 'Laser Printer'

where code=8;

1. Apply a 10% discount to all products.

-> update products

set price=price\*0.9;

1. Apply a 10% discount to all products with a price larger than or equal to $120.

-> update products

set price = price \* 0.9

where price >= 120;