**LIBRARY DATABASE**

BOOK (Book\_id, Title, Publisher\_Name, Pub\_Year)

BOOK\_AUTHORS (Book\_id, Author\_Name)

PUBLISHER (Name, Address, Phone)

BOOK\_COPIES (Book\_id, Programme\_id, No of\_Copies)

BOOK\_LENDING (Book\_id, Programme\_id, Card\_No, Date\_Out, Due\_Date)

LIBRARY\_PROGRAMME (Programme\_id, Programme\_Name, Address)

* **FIRST CREATE PUBLISHER TABLE SINCE BOOK TABLE HAS TO REFERENCE IT FOR PUBLISHER\_NAME.**
* **ALSO CREATE LIBRARY\_PROGRAMME FIRST SINCE BOOK\_COPIES AND BOOK\_LENDING BOTH HAVE REFERENCES TO PROGRAMME\_ID.**
* **IF MORE THAN 1 PRIMARY KEY SPECIFY AT THE END AS PRIMARY KEY(PK1,PK2,...)**

CREATE TABLE PUBLISHER (NAME VARCHAR2(20) PRIMARY KEY,PHONE INTEGER, ADDRESS VARCHAR2(20));

CREATE TABLE BOOK (BOOK\_ID INTEGER PRIMARY KEY, TITLE VARCHAR(20),PUB\_YEAR VARCHAR(20),PUBLISHER\_NAME REFERENCES PUBLISHER (NAME) ON DELETE CASCADE);

CREATE TABLE BOOK\_AUTHORS (AUTHOR\_NAME VARCHAR2(20),BOOK\_ID REFERENCES BOOK (BOOK\_ID) ON DELETE CASCADE);

CREATE TABLE LIBRARY\_PROGRAMME (PROGRAMME\_ID INTEGER PRIMARY KEY,PROGRAMME\_NAME VARCHAR2(50),ADDRESS VARCHAR2(50));

CREATE TABLE BOOK\_COPIES (BOOK\_ID REFERENCES BOOK (BOOK\_ID) ON DELETE CASCADE,PROGRAMME\_ID REFERENCES LIBRARY\_PROGRAMME (PROGRAMME\_ID) ON DELETE CASCADE,NO\_OF\_COPIES INTEGER, PRIMARY KEY (BOOK\_ID,PROGRAMME\_ID));

CREATE TABLE BOOK\_LENDING (BOOK\_ID REFERENCES BOOK (BOOK\_ID) ON DELETE CASCADE,PROGRAMME\_ID REFERENCES LIBRARY\_PROGRAMME (PROGRAMME\_ID) ON DELETE CASCADE,CARD\_NO INTEGER,DATE\_OUT DATE,DUE\_DATE DATE,PRIMARY KEY(BOOK\_ID,PROGRAMME\_ID,CARD\_NO));

**INSERTING VALUES**

**PUBLISHER**

INSERT INTO PUBLISHER VALUES ('JOHN',1234567890,'PARIS');

INSERT INTO PUBLISHER VALUES ('SAM',7411258963,'LONDON');

INSERT INTO PUBLISHER VALUES ('SMITH',4567891230,'BELGIUM');

**BOOK**

INSERT INTO BOOK VALUES (4,'THE EAST','JAN-2017','JOHN');

INSERT INTO BOOK VALUES (5,'PRINCESS','JUN-2017','SMITH');

INSERT INTO BOOK VALUES (7,'PRINCE','JUN-2019','SMITH');

INSERT INTO BOOK VALUES (8,'ROCK','JAN-2019','SAM');

**BOOK\_AUTHORS**

INSERT INTO BOOK\_AUTHORS VALUES ('BUNDLE',4);

INSERT INTO BOOK\_AUTHORS VALUES ('YOU',5);

INSERT INTO BOOK\_AUTHORS VALUES ('ME',7);

INSERT INTO BOOK\_AUTHORS VALUES ('ME',8);

**LIBRARY\_PROGRAMME**

INSERT INTO LIBRARY\_PROGRAMME VALUES (1,'CSE','PARIS');

INSERT INTO LIBRARY\_PROGRAMME VALUES (2,'ISE', 'LONDON');

INSERT INTO LIBRARY\_PROGRAMME VALUES (3,'ECE','BELGIUM');

**BOOK\_COPIES**

INSERT INTO BOOK\_COPIES VALUES (4,1,10);

INSERT INTO BOOK\_COPIES VALUES (5,1,10);

INSERT INTO BOOK\_COPIES VALUES (7,1,10);

INSERT INTO BOOK\_COPIES VALUES (4,2,20);

INSERT INTO BOOK\_COPIES VALUES (5,2,20);

INSERT INTO BOOK\_COPIES VALUES (7,2,20);

INSERT INTO BOOK\_COPIES VALUES (4,3,30);

INSERT INTO BOOK\_COPIES VALUES (5,3,30);

INSERT INTO BOOK\_COPIES VALUES (7,3,30);

INSERT INTO BOOK\_COPIES VALUES (8,3,30);

**BOOK\_LENDING**

INSERT INTO BOOK\_LENDING VALUES (4,1,500,'01-JUN-2017','07-JUN-2017');

INSERT INTO BOOK\_LENDING VALUES (5,1,500,'01-FEB-2017','07-MAY-2017');

INSERT INTO BOOK\_LENDING VALUES (7,2,500,'01-JAN-2017','07-JUN-2017');

INSERT INTO BOOK\_LENDING VALUES (8,1,500,'01-APR-2017','07-MAY-2017');

INSERT INTO BOOK\_LENDING VALUES (4,1,501,'01-JUN-2017','07-JUN-2017');

INSERT INTO BOOK\_LENDING VALUES (4,1,502,'01-JUN-2017','07-JUN-2017');

**Write SQL queries to**

1. Retrieve details of all books in the library – id, title, name of publisher, authors, number of copies in each programme, etc.

SELECT B.BOOK\_ID,B.PUBLISHER\_NAME,B.TITLE,A.AUTHOR\_NAME,C.NO\_OF\_COPIES,L.PROGRAMME\_ID

FROM BOOK B,BOOK\_COPIES C,BOOK\_AUTHORS A,LIBRARY\_PROGRAMME L

WHERE B.BOOK\_ID=A.BOOK\_ID AND A.BOOK\_ID= C.BOOK\_ID AND C.PROGRAMME\_ID= L.PROGRAMME\_ID;

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **BOOK\_ID** | **PUBLISHER\_NAME** | **TITLE** | **AUTHOR\_NAME** | **NO\_OF\_COPIES** | **PROGRAMME\_ID** |
| 4 | JOHN | THE EAST | BUNDLE | 10 | 1 |
| 5 | SMITH | PRINCESS | YOU | 10 | 1 |
| 7 | SMITH | PRINCE | ME | 10 | 1 |
| 4 | JOHN | THE EAST | BUNDLE | 20 | 2 |
| 5 | SMITH | PRINCESS | YOU | 20 | 2 |
| 7 | SMITH | PRINCE | ME | 20 | 2 |
| 4 | JOHN | THE EAST | BUNDLE | 30 | 3 |
| 5 | SMITH | PRINCESS | YOU | 30 | 3 |
| 7 | SMITH | PRINCE | ME | 30 | 3 |
| 8 | SAM | ROCK | ME | 30 | 3 |

10 rows selected.

2. Get the particulars of borrowers who have borrowed more than 3 books, but from Jan 2017 to Jun 2017

SELECT CARD\_NO

FROM BOOK\_LENDING

WHERE DATE\_OUT BETWEEN '01-JAN-2017' AND '01-JUN-2017' GROUP BY CARD\_NO HAVING COUNT(\*)>3;

|  |
| --- |
| **CARD\_NO** |
| 500 |

3. Delete a book in BOOK table. Update the contents of other tables to reflect this data manipulation operation.

**PERFROM THIS QUERY AT LAST**

DELETE FROM BOOK

WHERE BOOK\_ID=8;

1 row(s) deleted.

## Result Set 8

|  |  |  |  |
| --- | --- | --- | --- |
| **BOOK\_ID** | **TITLE** | **PUB\_YEAR** | **PUBLISHER\_NAME** |
| 4 | THE EAST | JAN-2017 | JOHN |
| 7 | PRINCE | JUN-2019 | SMITH |
| 5 | PRINCESS | JUN-2017 | SMITH |

3 rows selected.

4. Partition the BOOK table based on year of publication. Demonstrate its working with a simple query.

CREATE VIEW V\_PUBLICATION AS

SELECT PUB\_YEAR

FROM BOOK;

SELECT \* FROM V\_PUBLICATION;

## Result Set 5

|  |
| --- |
| **PUB\_YEAR** |
| JAN-2017 |
| JUN-2019 |
| JAN-2019 |
| JUN-2017 |

5. Create a view of all books and its number of copies that are currently available in the Library.

CREATE VIEW BOOK\_VIEW AS

SELECT B.BOOK\_ID,C.NO\_OF\_COPIES,B.TITLE,L.PROGRAMME\_ID

FROM BOOK B,BOOK\_COPIES C,LIBRARY\_PROGRAMME L

WHERE B.BOOK\_ID=C.BOOK\_ID AND C.PROGRAMME\_ID = L.PROGRAMME\_ID;

SELECT \* FROM BOOK\_VIEW;

## Result Set 7

|  |  |  |  |
| --- | --- | --- | --- |
| **BOOK\_ID** | **NO\_OF\_COPIES** | **TITLE** | **PROGRAMME\_ID** |
| 4 | 10 | THE EAST | 1 |
| 5 | 10 | PRINCESS | 1 |
| 7 | 10 | PRINCE | 1 |
| 4 | 20 | THE EAST | 2 |
| 5 | 20 | PRINCESS | 2 |
| 7 | 20 | PRINCE | 2 |
| 4 | 30 | THE EAST | 3 |
| 5 | 30 | PRINCESS | 3 |
| 7 | 30 | PRINCE | 3 |
| 8 | 30 | ROCK | 3 |

10 rows selected.

**ORDER DATABASE**

2) Consider the following schema for Order Database:

SALESMAN (Salesman\_id, Name, City, Commission)

CUSTOMER (Customer\_id, Cust\_Name, City, Grade,Salesman\_id)

ORDERS (Ord\_No, Purchase\_Amt, Ord\_Date, Customer\_id, Salesman\_id)

CREATE TABLE SALESMAN (SALESMAN\_ID INTEGER PRIMARY KEY,NAME VARCHAR2(50),CITY VARCHAR2(20),COMMISSION VARCHAR2(10));

CREATE TABLE CUSTOMER (CUSTOMER\_ID INTEGER PRIMARY KEY,CUST\_NAME VARCHAR2(20),CITY VARCHAR2(20),GRADE VARCHAR2(20),SALESMAN\_ID REFERENCES SALESMAN (SALESMAN\_ID) ON DELETE SET NULL);

CREATE TABLE ORDERS (ORD\_NO INTEGER PRIMARY KEY,PURCHASE\_AMT INTEGER,ORD\_DATE DATE,CUSTOMER\_ID REFERENCES CUSTOMER (CUSTOMER\_ID) ON DELETE CASCADE,SALESMAN\_ID REFERENCES SALESMAN (SALESMAN\_ID) ON DELETE CASCADE);

**INSERTING VALUES**

**SALESMAN**

INSERT INTO SALESMAN VALUES (101,'JOHN','BANGLORE','22%');

INSERT INTO SALESMAN VALUES (1000,'SMITH','MYSORE','30%');

INSERT INTO SALESMAN VALUES (1002,'BEN','BANGLORE','10%');

INSERT INTO SALESMAN VALUES (1003,'EVAN','DELHI','10%');

**CUSTOMER**

INSERT INTO CUSTOMER VALUES (1,'APRIL','BANGLORE',10,101);

INSERT INTO CUSTOMER VALUES (2,'JUNE','MYSORE',15,101);

INSERT INTO CUSTOMER VALUES (3,'CHRISTINE','BANGLORE',10,1002);

INSERT INTO CUSTOMER VALUES (4,'SAN','BANGLORE',5,1000);

**ORDERS**

INSERT INTO ORDERS VALUES (500,1000,'01-JAN-21',1,1000);

INSERT INTO ORDERS VALUES (501,100,'01-JUN-21',1,1002);

INSERT INTO ORDERS VALUES (502,500,'01-SEP-21',2,101);

INSERT INTO ORDERS VALUES (503,190,'01-JAN-21',3,101);

INSERT INTO ORDERS VALUES (504,700,'01-JUL-21',4,101);

**Write SQL queries to**

1. Count the customers with grades above Bangalore’s average.

SELECT GRADE,COUNT(CUSTOMER\_ID)

FROM CUSTOMER

GROUP BY GRADE HAVING GRADE > (SELECT AVG(GRADE) FROM CUSTOMER WHERE CITY='BANGLORE');

## Result Set 1

|  |  |
| --- | --- |
| **GRADE** | **COUNT(CUSTOMER\_ID)** |
| 15 | 1 |
| 10 | 2 |

2 rows selected.

2. Find the name and numbers of all salesmen who had more than one customer.

SELECT S.NAME,S.SALESMAN\_ID

FROM SALESMAN S

WHERE 1 < (SELECT COUNT (\*) FROM CUSTOMER C WHERE S.SALESMAN\_ID=C.SALESMAN\_ID);

## Result Set 6

|  |  |
| --- | --- |
| **NAME** | **SALESMAN\_ID** |
| JOHN | 101 |

3. List all salesmen and indicate those who have and don’t have customers in their cities (Use UNION operation.)

SELECT S.SALESMAN\_ID,S.NAME, C.CUST\_NAME, S.COMMISSION

FROM SALESMAN S, CUSTOMER C

WHERE S.CITY = C.CITY

UNION

SELECT S.SALESMAN\_ID, S.NAME, 'NO CUSTOMER', S.COMMISSION

FROM SALESMAN S

WHERE NOT CITY = ANY (SELECT CITY FROM CUSTOMER) ORDER BY 2 DESC;

## Result Set 4

|  |  |  |  |
| --- | --- | --- | --- |
| **SALESMAN\_ID** | **NAME** | **CUST\_NAME** | **COMMISSION** |
| 1000 | SMITH | JUNE | 30% |
| 101 | JOHN | APRIL | 22% |
| 101 | JOHN | CHRISTINE | 22% |
| 101 | JOHN | SAN | 22% |
| 1003 | EVAN | NO CUSTOMER | 10% |
| 1002 | BEN | APRIL | 10% |
| 1002 | BEN | CHRISTINE | 10% |
| 1002 | BEN | SAN | 10% |

[Download CSV](https://livesql.oracle.com/apex/f?p=590:1:147165667620::NO:RP::#)8 rows selected.

4. Create a view that finds the salesman who has the customer with the highest order of a day.

CREATE VIEW HIGHEST\_ORDER AS

SELECT O.ORD\_DATE,S.SALESMAN\_ID,S.NAME

FROM ORDERS O,SALESMAN S

WHERE S.SALESMAN\_ID=O.SALESMAN\_ID AND PURCHASE\_AMT =(SELECT MAX(PURCHASE\_AMT) FROM ORDERS O);

SELECT \* FROM HIGHEST\_ORDER;

## Result Set 2

|  |  |  |
| --- | --- | --- |
| **ORD\_DATE** | **SALESMAN\_ID** | **NAME** |
| 01-JAN-21 | 1000 | SMITH |

5. Demonstrate the DELETE operation by removing salesman with id 1000. All his orders must also be deleted.

DELETE FROM SALESMAN

WHERE SALESMAN\_ID=1000;

SELECT \* FROM SALESMAN;

## Result Set 7

|  |  |  |  |
| --- | --- | --- | --- |
| **SALESMAN\_ID** | **NAME** | **CITY** | **COMMISSION** |
| 101 | JOHN | BANGLORE | 22% |
| 1002 | BEN | BANGLORE | 10% |
| 1003 | EVAN | DELHI | 10% |

[Download CSV](https://livesql.oracle.com/apex/f?p=590:1:147165667620::NO:RP::#)3 rows selected.