**PART – A – Joining Data from Multiple Tables**

1. Create a list that displays the title of each book and the name and phone number of the contact at the publisher’s office for reordering each book.

We can write sql query in two ways

SELECT b.title, p.contact, p.phone

FROM books b, publisher p

WHERE b.pubid = p.pubid;

SELECT b.title, p.contact, p.phone

FROM books b JOIN publisher p

USING (pubid);

Graphical user interface, text, application, email

Description automatically generated

2. Determine which orders haven’t yet shipped and the name of the customer who placed the order. Sort the results by the date on which the order was placed.

SELECT c.firstname, c.lastname, o.order#

FROM customers c, orders o

WHERE c.customer# = o.customer#

AND o.shipdate IS NULL

ORDER BY o.orderdate;

SELECT c.firstname, c.lastname, o.order#

FROM customers c JOIN orders o

USING (customer#)

WHERE o.shipdate IS NULL

ORDER BY o.orderdate;

Graphical user interface, text, application

Description automatically generated

3. Which books were written by an author with the last name Adams? Perform the search using the author name.

SELECT b.title

FROM books b, bookauthor ba, author a

WHERE b.isbn = ba.isbn

AND ba.authorid = a.authorid

AND a.lname = 'ADAMS';

SELECT b.title

FROM books b JOIN bookauthor USING (isbn)

JOIN author a USING (authorid)

WHERE a.lname = 'ADAMS';

Graphical user interface, text, application, email

Description automatically generated

4. What gift will a customer who orders the book Shortest Poems receive? Use the actual book retail value to determine the gift.

SELECT p.gift

FROM books b, promotion p

WHERE b.retail BETWEEN p.minretail AND p.maxretail

AND b.title = 'SHORTEST POEMS';

SELECT p.gift

FROM books b JOIN promotion p

ON b.retail BETWEEN p.minretail AND p.maxretail

WHERE b.title = 'SHORTEST POEMS';

Graphical user interface, text, application, email

Description automatically generated

5. Display a list of all books in the BOOKS table. If a book has been ordered by a customer, also list the corresponding order number and the state in which the customer resides.

SELECT b.title, order#, c.state

FROM books b

LEFT OUTER JOIN orderitems oi USING (isbn)

LEFT OUTER JOIN orders USING (order#)

LEFT OUTER JOIN customers c USING (customer#);

Graphical user interface

Description automatically generated with medium confidence

Table

Description automatically generated

6. An EMPLOYEES table was added to the JustLee Books database to track employee information. Display a list of each employee’s name, job title, and manager’s name. Use column aliases to clearly identify employee and manager name values. Include all employees in the list and sort by manager name.

Creating Employee Table

CREATE TABLE EMPLOYEES

(

EMPID NUMBER,

EMPLOYEENAME VARCHAR2(20),

JOBTITLE VARCHAR2(20),

MANAGERID NUMBER NOT NULL,

SALARY VARCHAR2(20),

CONSTRAINT EMPLOYEES\_PK PRIMARY KEY( EMPID)

ENABLE

);

Graphical user interface, text, application, email

Description automatically generated

Then I have inserted the data into the table.

Table

Description automatically generated

SELECT e.employeename as "Employee Name", e.jobtitle,m.employeename "Manager Name"

FROM employees e LEFT OUTER JOIN employees m

ON e.managerid = m.empid

ORDER BY "Manager Name";

Graphical user interface, text, application, email

Description automatically generated

**PART B: Sequence and Index**

7. Create a sequence for populating the Customer# column of the CUSTOMERS table. When setting the start and increment values, keep in mind that data already exists in this table. The options should be set to not cycle the values and not cache any values, and no minimum or maximum values should be declared.

create sequence customers\_customer#\_seq

increment by 1

start with 101

nocycle

nocache

nominvalue

nomaxvalue;

Graphical user interface, text, application, email

Description automatically generated

8. Add a new customer row by using the sequence created in Question 1. The only data currently available for the customer is as follows: last name = Shoulders, first name = Frank, and zip = 23567.

insert into customers (customer#, lastname, firstname, zip)

Values(customers\_customer#\_seq.NEXTVAL, 'Shoulders', 'Frank', 23567);

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, application, Word

Description automatically generated

9. Create a sequence that generates integers starting with the value 5. Each value should be three less than the previous value generated. The lowest possible value should be 0, and the sequence shouldn’t be allowed to cycle. Name the sequence MY\_FIRST\_SEQ.

create sequence MY\_FIRST\_SEQ

increment by -3

start with 5

MINVALUE 0

MAXVALUE 5

NOCYCLE;

Graphical user interface, text, application, email

Description automatically generated

10. Issue a SELECT statement that displays NEXTVAL for MY\_FIRST\_SEQ three times. Because the value isn’t being placed in a table, use the DUAL table in the FROM clause of the SELECT statement. What causes the error on the third SELECT?

SELECT MY\_FIRST\_SEQ.NEXTVAL FROM DUAL;

SELECT MY\_FIRST\_SEQ.NEXTVAL FROM DUAL;

SELECT MY\_FIRST\_SEQ.NEXTVAL FROM DUAL;

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, Word

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

11. Create a bitmap index on the CUSTOMERS table to speed up queries that search for customers based on their state of residence. Verify that the index exists, and then delete the index.

CREATE BITMAP INDEX customer\_state\_index

ON customers(state);

SELECT index\_name

FROM user\_indexes;

DROP INDEX customer\_state\_index;

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, chat or text message, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

12. Create a B-tree index on the customer’s Lastname column. Verify that the index exists by querying the data dictionary. Remove the index from the database.

CREATE INDEX customers\_lastname\_index

ON customers(lastname);

SELECT index\_name

FROM user\_indexes;

DROP INDEX customers\_lastname\_index;

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

**PART C: SubQuery**

13. List the book title and retail price for all books with a retail price lower than the average retail price of all books sold by JustLee Books.

SELECT title, retail

FROM books

WHERE retail < (SELECT AVG(retail) FROM books);

Graphical user interface, text, application, email

Description automatically generated

14. Determine which books cost less than the average cost of other books in the same category.

SELECT b1.title, b2.category, b1.cost

FROM books b1, (SELECT category, AVG(cost) avgcost FROM books GROUP BY category) b2

WHERE b1.category = b2.category AND b1.cost < b2.avgcost;

Graphical user interface, text, application, email

Description automatically generated

15. Determine which orders were shipped to the same state as order 1014.

SELECT order#

FROM orders

WHERE shipstate = (SELECT shipstate FROM orders WHERE order# = 1014);

Graphical user interface, text, application, email

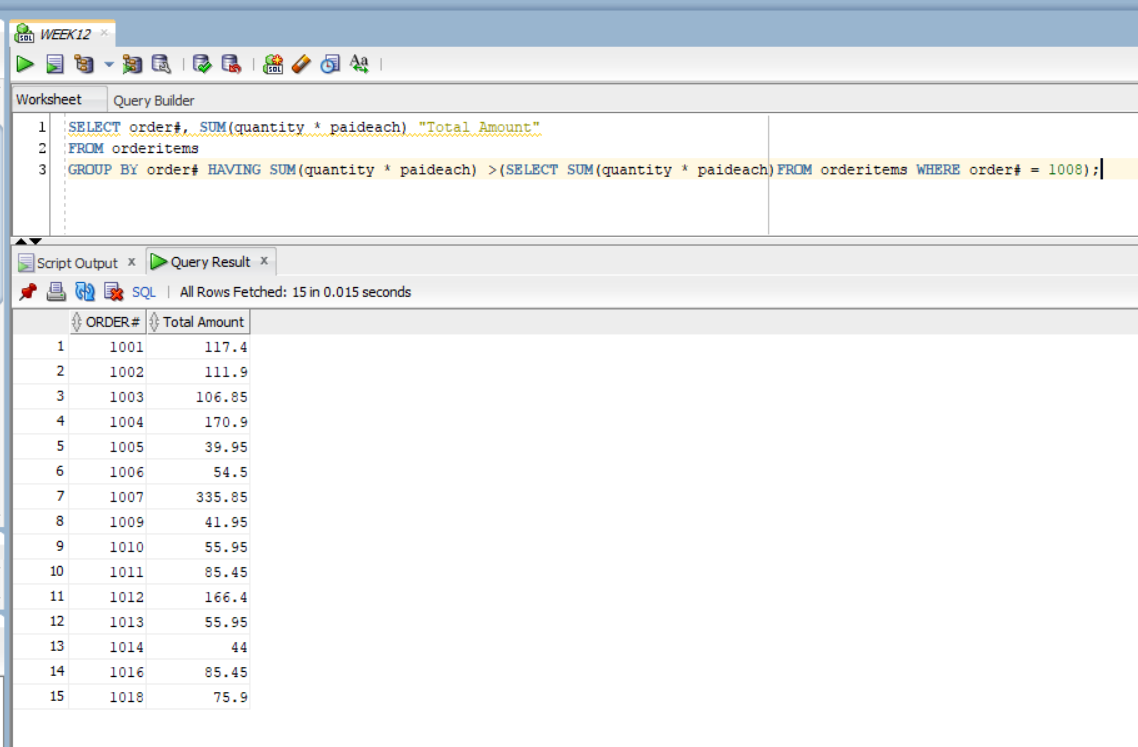
Description automatically generated

16. Determine which orders had a higher total amount due than order 1008

SELECT order#, SUM(quantity \* paideach) "Total Amount"

FROM orderitems

GROUP BY order# HAVING SUM(quantity \* paideach) >(SELECT SUM(quantity \* paideach)FROM orderitems WHERE order# = 1008);



17. Determine which author or authors wrote the books most frequently purchased by customers of JustLee Books.

SELECT lname, fname

FROM bookauthor JOIN author USING(authorid)

WHERE isbn IN

(SELECT isbn FROM orderitems GROUP BY isbn HAVING SUM(quantity) = (SELECT MAX(COUNT(\*)) FROM orderitems GROUP BY isbn));

Graphical user interface, text, application

Description automatically generated