Alan Moss

CSCD 327

Lab 6

1) Determine which orders were shipped to the same state as order 1000. Note: it is okay to include order 1000 in the result

```
1
2   SELECT
3       o.Order_num
4   FROM
5       ORDERS o
6   WHERE
7       ShipState = (SELECT ShipState
8                    FROM ORDERS
9                    WHERE Order_num = 1000)
10  ;
```

Order_num
1000
1009

2) List the shipping city and state for the order that had the shortest shipping delay. Note: shipping delay = datediff(shipdate, orderdate)

```
1
2   SELECT
3       ShipCity, ShipState, datediff(ShipDate, OrderDate) as "ship_delay"
4   FROM
5       ORDERS
6   WHERE
7       datediff(ShipDate, OrderDate) = (SELECT MIN(datediff(ShipDate, OrderDate))
8                                        FROM ORDERS)
9   ;
```

ShipCity	ShipState	ship_delay
EASTPOINT	FL	0

3) List the book title and retail price for books with a retail price greater than the average PaidEach price of that book.

```
1
2 •  SELECT DISTINCT
3        Title, Retail
4    FROM
5        BOOKS b, ORDERITEMS oi
6    WHERE
7        b.ISBN = oi.ISBN
8    AND
9        b.Retail > (SELECT AVG(oi.PaidEach) FROM ORDERITEMS)
10   ;
```

Title	Retail
PAINLESS CHILD-REARING	89.95
HOLY GRAIL OF ORACLE	75.95

4) Determine which books cost less than the average cost of the books in the same category. List the title, the category, and the cost of these books

```
1
2 •  SELECT
3        b.Title, b2.Category, b.Cost
4    FROM
5        BOOKS b, (SELECT Category, AVG(Cost) AS average
6                  FROM BOOKS GROUP BY Category) b2
7    WHERE
8        b.Category = b2.Category
9    AND
10       b.Cost < b2.average
11   ;
```

Title	Category	Cost
REVENGE OF MICKEY	FAMILY LIFE	14.20
HANDCRANKED COMPUTERS	COMPUTER	21.80
COOKING WITH MUSHROOMS	COOKING	12.50
BIG BEAR AND LITTLE DOVE	CHILDREN	5.32
DATABASE IMPLEMENTATION	COMPUTER	31.40

5) Determine which customers placed orders for the most expensive book (in terms of regular retail price) carried by the bookstore.

```sql
SELECT
    Customer_num
FROM
    CUSTOMERS JOIN ORDERS USING (Customer_num)
    JOIN ORDERITEMS USING(Order_num)
    JOIN BOOKS USING(ISBN)
WHERE
    Retail = (SELECT MAX(Retail) FROM BOOKS)
;
```

Customer_num
1010
1020
1010
1017
1003

6) Determine which orders had a total order amount greater than or equal to order 1012's total order amount. [Note: total order amount = sum(Quantity * PaidEach)]

```sql
SELECT
    Order_num, SUM(Quantity * PaidEach) Total_Order_Amount
FROM
    ORDERITEMS
GROUP BY
    Order_num
HAVING
    SUM(Quantity * PaidEach) >= (SELECT SUM(Quantity * PaidEach)
                                 FROM ORDERITEMS
                                 WHERE Order_num = 1012)
;
```

Order_num	Total_Order_Amount
1004	170.90
1007	335.85
1012	166.40

7) List the title of all books in the same categories as books previously purchased by customer 1001. Don't include books this customer has already purchased.

```sql
SELECT
    Title
FROM
    BOOKS
WHERE
    Category
IN
    (SELECT DISTINCT
        Category
    FROM
        BOOKS JOIN ORDERITEMS USING (ISBN)
        JOIN ORDERS USING (Order_num)
    WHERE
        Customer_num = 1001)
AND
    ISBN not in
    (SELECT ISBN FROM ORDERS JOIN ORDERITEMS USING (Order_num)
     WHERE Customer_num = 1001)
;
```

Title
THE WOK WAY TO COOK
HANDCRANKED COMPUTERS
HOLY GRAIL OF ORACLE
E-BUSINESS THE EASY WAY

8) Find the customer who has the highest total order amount. [Note: total order amount = sum(Quantity * PaidEach)]

```sql
SELECT Customer_num, LastName, FirstName, SUM(Quantity * PaidEach) Total_Order_Amount
FROM
    ORDERITEMS NATURAL JOIN ORDERS
    NATURAL JOIN CUSTOMERS
GROUP BY
    Customer_num
HAVING
    Total_Order_Amount = (SELECT MAX(cust.total_amt)
                    FROM (SELECT SUM(Quantity * PaidEach) as total_amt
                    FROM ORDERITEMS NATURAL JOIN ORDERS
                    NATURAL JOIN CUSTOMERS
                    GROUP BY Customer_num) as cust)
;
```

Customer_num	LastName	FirstName	Total_Order_Amount
1007	GIANA	TAMMY	379.85

9) Determine the number of different customers who have placed an order for books written or co-written by JACK BAKER.

```sql
SELECT COUNT(DISTINCT customer_num) Num_of_Customers
FROM
    ORDERS JOIN ORDERITEMS USING (Order_num)
WHERE
    ISBN
IN
    (SELECT ISBN
    FROM ORDERITEMS JOIN BOOKAUTHOR USING (ISBN)
        JOIN AUTHOR USING (AuthorID)
    WHERE
        Fname = 'JACK'
    AND
        Lname = 'BAKER')
;
```

Num_of_Customers
8

10) Find the instructor earning the highest salary. (Don't use ORDER BY and LIMIT in your solution.)

```sql
SELECT
    ID, name, MAX(salary) salary
FROM
    instructor
WHERE
    salary >= (SELECT MAX(salary) FROM instructor)
GROUP BY
    ID
;
```

ID	name	salary
22222	Einstein	95000.00

11) Find the instructor earning the second-lowest salary. (Don't use ORDER BY and LIMIT in your solution.)

```sql
SELECT
    ID, name, MIN(salary) salary
FROM
    instructor
WHERE
    salary = (SELECT MIN(salary)
              FROM instructor
              WHERE salary > (SELECT MIN(salary) from instructor))
GROUP BY
    ID
;
```

ID	name	salary
32343	El Said	60000.00

12) Find the student who has taken the most number of distinct courses. (Don't use ORDER BY and LIMIT in your solution.)

13) Find the student who has earned the most total credits. Important Note: The total credits a student has earned should be calculated based on the courses the student has taken and the grades the student has received. Don't include the courses that the student didn't pass, i.e., Grade is null or Grade = 'F'. The tot_cred field in the student relation has the incorrect information about the total credits. It was there on purpose for the topic of database update, so don't use this field for this query).

```sql
SELECT
    student.ID, name, SUM(credits) total_credit
FROM
    student JOIN takes USING (ID)
            JOIN course USING (course_id)
WHERE
     grade <> 'F'
GROUP BY
    student.ID
;
```

ID	name	total_credit
00128	Zhang	7
12345	Shankar	14
19991	Brandt	3
23121	Chavez	3
44553	Peltier	4
45678	Levy	7
54321	Williams	8
55739	Sanchez	3
76543	Brown	7
76653	Aoi	3
98765	Bourikas	7
98988	Tanaka	4