ACIT 1630
Relational Database Design and SQL

Review Questions
Lesson 11

Answer each of the following questions labeling your answers clearly. Save your work in the Desire2Learn **Drop box** for **Lesson 08**.

Part A:

1. What is the difference between UNION and UNION ALL?

   UNION keeps unique records, UNIION ALL keeps all records, even duplicate values.

2. Suppose that you have two tables, EMPLOYEE and EMPLOYEE_1. The EMPLOYEE table contains the records for four employees: Alice Cordoza, John Cretchakov, Susan Harrison, and Anne McDonald. The EMPLOYEE_1 table contains the records for two employees: John Cretchakov and Mary Chen. Given that information, list the query output for the UNION query.

The output is:
Alice Cordoza
John Cretchakov
Susan Harrison
Anne McDonald
Mary Chen

3. Given the employee information in question 2 above, list the query output for the UNION ALL query.

The output is:
Alice Cordoza
John Cretchakov
Susan Harrison
Anne McDonald
John Cretchakov
Mary Chen

4. What is the difference between an inner join and an outer join?

INNER JOIN selects records that have matching pairs in both tables, when given criteria.
OUTER JOIN selects records that have matching pairs in both tables but also keeps the unmatched pairs.

5. Suppose that a PRODUCT table contains two attributes, PROD_CODE and VEND_CODE. Those two attributes have values of ABC, 125, DEF, 124, GHI, 124, and JKL, 123, respectively. The VENDOR table contains a single attribute, VEND_CODE, with values 123, 124, 125, and 126, respectively. The VEND_CODE attribute in the PRODUCT table is a foreign key to the VEND_CODE in the VENDOR table. Given that information, what would be the query output for a UNION query based on these two tables? (Because the common attribute is V_CODE, the output will only show the V_CODE values generated by the each query.)

It would be 125, 124, 123, 126. Duplicates won't be shown

6. Suppose that a PRODUCT table contains two attributes, PROD_CODE and VEND_CODE. Those two attributes have values of ABC, 125, DEF, 124, GHI, 124, and JKL, 123, respectively. The VENDOR table contains a single attribute, VEND_CODE, with values 123, 124, 125, and 126, respectively. The VEND_CODE attribute in the PRODUCT table is a foreign key to the VEND_CODE in the VENDOR table. Given that information, what would be the query output for a UNION ALL query based on these two tables? (Because the common attribute is V_CODE, the output will only show the V_CODE values generated by the each query.)

It would be: 125,124,124,123,123,124,125,126. Duplicates included.

7. What are the three join types included in the OUTER JOIN classification? Describe each of the types.

LEFT OUTER JOIN will yield all the matching rows in the join columns, including the unmatched rows in the Left(first) table.
RIGHT OUTER JOIN will yield all the matching rows in the join columns, including the unmatched rows in the Right(second) table.
FULL OUTER JOIN will yield all the matching rows in the join columns, including all the unmatched rows from both tables.

8. What string function should you use to list the first three characters of a company's EMP_LNAME values? Give an example, using a table named EMPLOYEE.

```
SELECT SUBSTRING(EMP_LNAME, 1, 3)
FROM EMPLOYEE
```

9. Describe a SQL batch.

It is a set of SQL commands executed together as a group.

Table name: CUSTOMER

CUST_NUM	CUST_LNAME	CUST_FNAME	CUST_BALANCE
1000	Smith	Jeanne	1050.11
1001	Ortega	Juan	840.92

Table name: CUSTOMER_2

CUST_NUM	CUST_LNAME	CUST_FNAME
2000	McPherson	Anne
2001	Ortega	Juan
2002	Kowalski	Jan
2003	Chen	George

Table name: INVOICE

INV_NUM	CUST_NUM	INV_DATE	INV_AMOUNT
8000	1000	23-Mar-10	235.89
8001	1001	23-Mar-10	312.82
8002	1001	30-Mar-10	528.10
8003	1000	12-Apr-10	194.78
8004	1000	23-Apr-10	619.44

10. Using the INVOICE table shown above, write the query that will show the invoice number, the invoice amount, and the average invoice amount from the INVOICE table.

```
select inv_num, inv_amount, avg(inv_amount)
from INVOICE
```

11. Using the CUSTOMER table shown above, modify the CUSTOMER table to included two new attributes: CUST_DOB and CUST_AGE.

```
alter table CUSTOMER
add CUST_DOB date,
CUST_AGE int
```

12. Using the CUSTOMER table shown above, modify the birth date for customer 1000 to March 15, 1979, and customer 1001 to December 22, 1988.

```
UPDATE CUSTOMER
SET CUST_DOB = '1979-03-15'
WHERE CUST_NUM = 1000

UPDATE CUSTOMER
SET CUST_DOB = '1988-12-22'
WHERE CUST_NUM = 1000
```

13. Explain why the two following commands produce different results.

    SELECT DISTINCT COUNT (V_CODE) FROM PRODUCT;
    SELECT COUNT (DISTINCT V_CODE) FROM PRODUCT;

In the first query we get the count of V_CODE, and then we get the DISTINCT value, whereas the second query we take the DISTINCT value of V_CODE and THEN the Count Value of V_CODE.

14. Explain the difference between an ORDER BY clause and a GROUP BY clause.

GROUP BY is used to group results according to distinct values of a selected column.
ORDER BY sorts the rows in ascending or descending order, according to the selected column values.

15. Using the INVOICE table shown above, write the query to show the number of invoices that have invoice amount over $300.

```
Select * From INVOICE
Where INV_AMOUNT > 300
order by ASC
```

16. In a SELECT query, what is the difference between a WHERE clause and a HAVING clause?

The WHERE clause yields rows that match given constraints/conditions.
The HAVING clause will filter results based on an aggregate function.

EMPLOYEE

EMP_NUM	EMP_LNAME	EMP_FNAME	EMP_INITIAL	EMP_HIREDATE	JOB_CODE
101	News	John	G	08-Nov-00	502
102	Senior	David	H	12-Jul-89	501
103	Arbough	June	E	01-Dec-96	500
104	Ramoras	Anne	K	15-Nov-87	501
105	Johnson	Alice	K	01-Feb-93	502
106	Smithfield	William		22-Jun-04	500
107	Alonzo	Maria	D	10-Oct-93	500
108	Washington	Ralph	B	22-Aug-91	501
109	Smith	Larry	W	18-Jul-97	501

17. Using the EMPLOYEE table shown above, write the SQL code to insert the rows for employee numbers 106 and 107 for the table.

```
INSERT INTO EMPLOYEE (EMP_NUM, EMP_LNAME, EMP_FNAME, EMP_INITIAL,
EMP_HIREDATE, JOB_CODE) VALUES
(105, 'Johnson', 'Alice', 'K', '01-Feb-93', 502),
(106, 'Smithfield', 'William', null, '22-Jun-04', 500);
```

18. Using the EMPLOYEE table shown above, write the SQL code to change the job code to 500 for the person whose employee number is 101.

```
UPDATE EMPLOYEE
SET JOB_CODE = 500
WHERE EMP_NUM = 101
```

19. Using the EMPLOYEE table shown above, write the SQL code to delete the row for the person named William Smithfield, who was hired on June 22, 2004, and whose job code classification is 500.

```
DELETE FROM EMPLOYEE
WHERE (EMP_FNAME ='William' and EMP_LNAME = 'Smithfield' and JOB_CODE =
500)
```

20. Using the EMPLOYEE table shown above, write the SQL code to change the job classification (JOB_CODE) to 502 for all employees whose job classification (JOB_CODE) is 501.

```
UPDATE EMPLOYEE
SET JOB_CODE = 502
WHERE JOB_CODE = 501
```

Part B:

1. List the title from the titles table, the order number and order date from the sales table, and the store name from the stores table. Display only the first 30 characters of the title column. There should be a row produced in the result set for each row in the titles table. Order the result set by the order number. The query should produce the result set listed below. (Hint: LEFT OUTER JOIN statement)

title	ord_num	ord_date	stor_name
Net Etiquette	NULL	NULL	NULL
The Psychology of Computer Coo	NULL	NULL	NULL
The Gourmet Microwave	423LL922	1994-09-14 00:00:00.000	Bookbeat
.....			
Straight Talk About Computers	QQ2299	1993-10-28 00:00:00.000	Fricative Bookshop
Silicon Valley Gastronomic Tre	TQ456	1993-12-12 00:00:00.000	Fricative Bookshop
You Can Combat Computer Stress	X999	1993-02-21 00:00:00.000	Fricative Bookshop

(23 row(s) affected)

```
select substring(title, 1, 30) as title, ord_num, ord_date, stor_name
from titles
left outer join sales on titles.title_id = sales.title_id
left outer join stores on sales.stor_id = stores.stor_id
order by ord_num
```

2. Using the INSERT command, add two books to the titles table. The first book has a title id of ZZ1234, title of Microsoft SQL Server, type of computer, publisher id of 0877, price of $89.99, and a publish date of September 29, 2008. The second book has a title id of TT2345, title of Designing Databases, type of computer, publisher id of 1389, priced of $168.98, and a publish date of January 1, 2009.

```
insert into titles (title_id, title, type, pub_id, price, pubdate) values
('ZZ1234','Microsoft SQL Server','computer','0877',89.99, '2008-09-29'),
('TT2345','Designing Databases','computer','1389',168.98, '2009-01-01')
```

3. List the sum of the order cost (quantity * price) for each order by store id, store name, order number, quantity, title, and price where the sum of the order cost is between $150.00 and $500.00. Display the order number, store id, and quantity from the sales table, the store name from the stores table, the price from the titles table, and the sum of the order cost. The query should produce the result set listed below.

ord_num	stor_id	stor_name	qty	price	order_cost
P2121	7067	News & Brews	20	11.95	239.00
P2121	7067	News & Brews	20	14.99	299.80
N914008	7131	Doc-U-Mat: Quality Laundry and Books	20	10.95	219.00
…					

TQ456	7896	Fricative Bookshop	10	19.99	199.90
423LL930	8042	Bookbeat	10	19.99	199.90
P723	8042	Bookbeat	25	11.95	298.75

(11 row(s) affected)

```
select sales.ord_num, sales.stor_id, stores.stor_name, sales.qty,
titles.price, (sales.qty * titles.price) as order_cost
from sales
left outer join titles on titles.title_id = sales.title_id
left OUTER join stores on sales.stor_id = stores.stor_id
WHERE ((sales.qty*titles.price) > 150.00) AND ((sales.qty*titles.price) <
500.00)
```

4. Using the UPDATE command, increase the price by 10% for the book Microsoft SQL Server with a title id of ZZ1234.

```
UPDATE titles
SET price = (price * 1.1)
WHERE title_id = 'ZZ1234'
```

5. Create a new table called BusinessBooks containing the title id, title, and price columns along with the data from the titles table which have a type of business. There should be 4 rows inserted into the new table.

```
select title_id, title, price, type
into BusinessBooks
from titles
where type = 'business'
```

6. Delete the books with a title id of ZZ1234 and TT2345 from the titles.

```
delete from titles
where (title_id = 'ZZ1234') or (title_id = 'TT2345')
```

7. List the average price and the sum of the price of the books for each type and the running total of all book prices from the titles table. The query should produce the result set listed below.

```
SELECT title, type, price,
AVG(price) OVER (PARTITION BY type) AS 'Average Price By Type',
SUM(price) OVER (PARTITION BY type) AS 'Total Price by Type',
SUM(price) OVER (ORDER BY type) AS 'Running Total Overall'
FROM titles
```

	title	type	price	Average Price By Type	Total Price By Type	Running Total Overall
1	The Busy Executive's Database Guide	business	19.99	13.73	54.92	54.92
2	Cooking with Computers: Surreptitious Balance Sheets	business	11.95	13.73	54.92	54.92
3	You Can Combat Computer Stress!	business	2.99	13.73	54.92	54.92
4	Straight Talk About Computers	business	19.99	13.73	54.92	54.92
5	Silicon Valley Gastronomic Treats	mod_cook	19.99	11.49	22.98	77.90
6	The Gourmet Microwave	mod_cook	2.99	11.49	22.98	77.90
7	But Is It User Friendly?	popular_comp	22.95	21.475	42.95	120.85
8	Secrets of Silicon Valley	popular_comp	20.00	21.475	42.95	120.85
9	Net Etiquette	popular_comp	NULL	21.475	42.95	120.85
10	Computer Phobic AND Non-Phobic Individuals: Beha...	psychology	21.59	13.504	67.52	188.37
11	Is Anger the Enemy?	psychology	10.95	13.504	67.52	188.37
12	Life Without Fear	psychology	7.00	13.504	67.52	188.37
13	Prolonged Data Deprivation: Four Case Studies	psychology	19.99	13.504	67.52	188.37
14	Emotional Security: A New Algorithm	psychology	7.99	13.504	67.52	188.37
15	Onions, Leeks, and Garlic: Cooking Secrets of the M...	trad_cook	20.95	15.9633	47.89	236.26
16	Fifty Years in Buckingham Palace Kitchens	trad_cook	11.95	15.9633	47.89	236.26
17	Sushi, Anyone?	trad_cook	14.99	15.9633	47.89	236.26
18	The Psychology of Computer Cooking	UNDECIDED	NULL	NULL	NULL	236.26

8. Using the UNION command, list the authors with a state of CA and the publishers with a state that does not contain NULL values. Display the author id, first name, last name, city, and state from the authors table, and the publisher id, publisher name, city, and state from the publishers table. Format the name of the author as the first name followed by a space followed by the last name. Order the result set by the four column - state. The query should produce the result set listed below.

ID	Name	city	state
1389	Algodata Infosystems	Berkeley	CA
172-32-1176	JohnsonWhite	Menlo Park	CA
213-46-8915	MarjorieGreen	Oakland	CA
.....			
0736	New Moon Books	Boston	MA
9952	Scootney Books	New York	NY
1756	Ramona Publishers	Dallas	TX

(21 row(s) affected)

```
SELECT au_ID, concat(au_fname,' ',au_lname) AS Name, city, state
FROM authors
WHERE state = 'CA'
UNION
SELECT pub_id, pub_name, city, state
FROM publishers
WHERE state IS NOT NULL
```

```
ORDER BY state
```

9. List the total of books by the type and publisher name. Display the type and minimum price from the titles table, the publisher name from the publishers table, and the sum of the quantity from the sales table. (Hint: Use a GROUP BY statement.)

type	pub_name	quantity	price
business	Algodata Infosystems	55	11.95
popular_comp	Algodata Infosystems	80	20.00
mod_cook	Binnet & Hardley	50	2.99
psychology	Binnet & Hardley	20	21.59
trad_cook	Binnet & Hardley	80	11.95
business	New Moon Books	35	2.99
psychology	New Moon Books	173	7.00

(7 row(s) affected)

```
SELECT type, pub_name, SUM(qty), MIN(price)
FROM titles
JOIN publishers ON titles.pub_id = publishers.pub_id
JOIN sales ON titles.title_id = sales.title_id
GROUP BY type, pub_name
```

10. List the sum of the quantity for each order by store id, order number, and order date where the sum of the quantity is between 10 and 50. Display the store id, order number, order date, and the sum of the quantity from the sales table. The query should produce the result set listed below.

stor_id	ord_num	ord_date	sum_quantity
7066	A2976	1993-05-24 00:00:00.000	50
7067	D4482	1994-09-14 00:00:00.000	10
7131	N914008	1994-09-14 00:00:00.000	20
.....			
8042	423LL930	1994-09-14 00:00:00.000	10
8042	P723	1993-03-11 00:00:00.000	25
8042	QA879.1	1993-05-22 00:00:00.000	30

(11 row(s) affected)

```
select stor_id, ord_num, ord_date, sum(qty) as sum_quantity
from sales
group by stor_id, ord_date, ord_num
having sum(qty) BETWEEN 10 and 50
```