**SQL - Data Manipulation**  
**- Subqueries**

*CIT 381*

**Part 1: DML - data manipulation language**

That part of SQL that retrieves or changes data in a database. In addition to the SELECT statement, there are INSERT, DELETE, and UPDATE statements.

**insert**

To insert a single row into the dept_locations table (of the company database).

```
INSERT INTO dept_locations
VALUES(1, 'Austin')
```

The above assumes you know the implicit order of the attributes. If not known, you can list the attribute names to be safe.

```
INSERT INTO dept_locations(dlocation, dnumber)
VALUES('Austin', 5)
```

You may also insert several rows at a time. (This may be unique to MySQL.)

```
INSERT INTO dept_locations
VALUES(1, 'Austin'),
       (5, 'Austin')
```

**update**

The following will reverse the names of all cities (dlocation) for department number 5.

```
UPDATE dept_locations
SET dlocation = REVERSE(dlocation)
WHERE dnumber=5
```

The SET clause can modify several attributes. The WHERE clause can be quite complicated - say by using a subquery.

**deletion**

This statement deletes entire rows - one or more.

```
DELETE FROM dept_locations
WHERE dlocation LIKE '%Austin%'
```

Be careful though.

```
DELETE FROM dept_locations
```

This will delete all rows from the table.
Part 2: Subqueries

Some simple joins:

Let's find customers who have a (current) rental. Simple with a join.

```
SELECT c.firstname, c.lastname
FROM customer c INNER JOIN rental r USING(accountid)
```

To get rid of multiple names, use DISTINCT.

```
SELECT DISTINCT c.firstname, c.lastname
FROM customer c INNER JOIN rental r USING(accountid)
```

What about those without a rental? Do an outer join, then look for a NULL.

```
SELECT DISTINCT c.firstname, c.lastname
FROM customer c LEFT OUTER JOIN rental r USING(accountid)
WHERE r.videoid IS NULL
```

Replace joins with subqueries:

A subquery is a SELECT statement in the WHERE clause. To find customers with rentals, think of the accountid values from the rental table as a set (in the mathematical sense).

```
SELECT c.accountid, c.firstname, c.lastname
FROM customer c
WHERE c.accountid IN (SELECT r.accountid FROM rental r)
```

Now it is easy to find customers (or rather, accounts) with no rental.

```
SELECT c.accountid, c.firstname, c.lastname
FROM customer c
WHERE c.accountid NOT IN (SELECT r.accountid FROM rental r)
```

Note that a customer can have several accounts. One person can have an account with a rental, and another with no rental. How would we find people who have no account with a rental? Their name should not appear in a list of names of people with rentals.

```
SELECT c.firstname, c.lastname
FROM customer c
WHERE (c.firstname, c.lastname) NOT IN
    (SELECT c.firstname, c.lastname
     FROM customer c INNER JOIN rental r USING(accountid))
```

Note: it is generally preferred to write a query with a join rather than a subquery, if possible. Query optimizers have many tricks with which to deal with joins, with subqueries not so much.

Set operations:

- IN
- NOT IN
- ALL
- ANY
- SOME
- EXISTS
- NOT EXISTS
People with accounts having a rental -

```sql
SELECT c.accountid, c.firstname, c.lastname
FROM customer c
WHERE c.accountid = SOME (SELECT r.accountid FROM rental r)
```

People with accounts with no rental -

```sql
SELECT c.accountid, c.firstname, c.lastname
FROM customer c
WHERE c.accountid <> ANY (SELECT r.accountid FROM rental r)
```

Find name of employee with highest hourly rate

```sql
SELECT e.firstname, e.lastname, h.hourlyrate
FROM employee e INNER JOIN hourlyemployee h USING(ssn)
WHERE h.hourlyrate >= ALL (SELECT hourlyrate FROM hourlyemployee)
```

Equivalently

```sql
SELECT e.firstname, e.lastname, h.hourlyrate
FROM employee e INNER JOIN hourlyemployee h USING(ssn)
WHERE h.hourlyrate = (SELECT MAX(hourlyrate) FROM hourlyemployee)
```

Watch for NULL!

Look at employee table of company database. Find names of people who are supervisors of someone, using a subquery.

```sql
SELECT fname, lname, ssn
FROM employee
WHERE ssn IN
  (SELECT superssn
   FROM employee)
```

How about non-supervisors? Try

```sql
SELECT fname, lname, ssn
FROM employee
WHERE ssn NOT IN
  (SELECT superssn
   FROM employee)
```

It fails because of the NULL superssn.

```sql
SELECT fname, lname, ssn
FROM employee
WHERE ssn NOT IN
  (SELECT superssn
   FROM employee
   WHERE superssn IS NOT NULL)
```

**uncorrelated subquery**

All the examples of subqueries above can be executed alone. These are uncorrelated.
**correlated subquery**

On the other hand, a subquery can reference the outside query. Look at the problem of finding accounts with no rentals.

```sql
SELECT c.accountid, c.firstname, c.lastname
FROM customer c
WHERE NOT EXISTS
  (SELECT *
   FROM rental r
   WHERE r.accountid=c.accountid)
```

The inner query refers to a value (c.accountid) from the outer query. Hence, it has to be executed for each c.accountid value. In general, correlated subqueries are less efficient.

**subquery in the FROM clause**

We can create a temporary table by putting a SELECT in the FROM clause. We must rename it.

Add up the total amount spent by each customer. Must look in the rental and previous rental tables.

```sql
SELECT c.firstname, c.lastname, SUM(s.cost) AS totspent
FROM customer c INNER JOIN
  (SELECT accountid, cost FROM rental
   UNION
   SELECT accountid, cost FROM previousrental) AS s
  USING(accountid)
GROUP BY 1,2
```