I. Review

[1] SQL Clause General Form

(1)	SELECT [DISTINCT] fields	(f)
(2)	FROM tables	(a)
(3)	[WHERE predicate]	(b)
(4)	[GROUP BY fields]	(c)
(5)	[HAVING predicate]	(d)
(6)	[ORDER BY fields] [ASC/DESC]	(e)

[1.1] Order of Clauses

. Clauses must be in the order of (1),(2),(3),(4),(5),(6).

. Only the first two are required.

. Usually, we cannot use HAVING clause without a GROUP BY clause.

[1.2] Order of Operations

. Output of one clause is the input to the next one in the order of (a),(b),(c),(d), (e), (f).

[2] SELECT

. Select specific list of columns from a table.

. Use an asterisk (*) to select all the columns from the table.

*e.g., SELECT * FROM Staff*

. Select keeps duplicates. Use the DISTINCT option to eliminate duplicated rows.

e.g., SELECT DISTINCT dept, job FROM Staff

. If the column name is not specified, the system will generate a number.

e.g., SELECT dept, name, salary+500 FROM Staff

Output: *dept* *name* *3*

xxx *xxx* *xxx*

. Use the options AS clause to assign a meaningful name.

e.g., SELECT dept, name, salary+500 AS pay FROM Staff

Output: *dept* *name* *pay*

xxx *xxx* *xxx*

[3] FROM

. List tables to join

. Use correlation name (range variables, tuple variables)

-- A correlation name can be defined in the FROM clause of a query to distinguish multiple uses of an object.

e.g., SELECT name, deptName

FROM Staff [AS] S, Org [AS] O

WHERE O.manager = S.id

Here, S and O are correlation names for Staff and Org, respectively.
The keyword AS is optional.

-- Once you have defined a correlation name, you can only use the correction name to qualify the object. Org.manager = Staff.id will fail.

-- It is userful to compare entries of a table with ifself.

e.g., SELECT e2.firstname, e2.lastname, e1.firstname, e1.lastname
FROM Employee e1, Employee e2
WHERE e1.workdept = e2.workdept .....

. Use nested table expression (subqueries)

-- A temporary view where the definition is nested in the FROM clause of the query.

```
e.g.,  SELECT name
       FROM MovieExec, (SELECT ProducerC#
                        FROM Movie, StarsIn
                        WHERE title = movidTitle
                        AND year = movidYear
                        AND starname = 'HARRISON Ford') Prod
       WHERE cert# = Prod.producerC#
```

[4] WHERE

. Select specific rows from a table.

. Specify a condition consisting of one or more predicates.

. Use a subquery

```
e.g., SELECT division, location
      FROM Org
      WHERE deptnumb = (SELECT dept
                        FROM Staff
                        WHERE id = 280)
```

Here, a subquery is first evaluated, once and for all, and the result is used in the high-level query.

. Use a correlated subquery

```
e.g., SELECT title
      FROM Movie Old
      WHERE year < ANY (SELECT year
                        FROM Movie
                        WHERE title = Old.title)
```

Here, a subquery will be evaluated many times, once for each assignment of a value to some term in the subquery that comes from a tuple variable outside the subquery.

[5] GROUP BY

. Group rows according to the group (consisting of columns called grouping columns) defined in a GROUP BY clause.

. The column names in the SELECT clause must be either grouping columns or aggregate operations(AVG, COUNT, MAX, MIN, SUM).

e.g., SELECT workdept, edlevel, max(salary) AS maxmum
FROM Employee
WHERE hiredata > '1979-01-01'
GROUP BY workdept, edlevel

[6] HAVING

. Apply a qualifying condition to groups so that the system returns a result only for the groups that satisfy the condition.

. An aggregation in a HAVING clause applies only to the tuples of the group being tested.

. Any attribute of relations in the FROM clause may be aggregated in the HAVING clause. But only those attributes that are in the GROUP BY list may appear unaggregated in the HAVING clause.

e.g., SELECT studioName, SUM(length)
FROM Movie
GROUP BY studioName
HAVING MIN(year) < 1930

(If non-grouping columns are appeared in the SELECT and/or HAVING caluses, they must be aggregated.)

[7] ORDER BY

. Sort the rows by the values in one or more columns.

. The column names in the ORDER BY clause do not have to be specified in the select list.

. Use ASC (default) or DESC to order rows in ascending or descending order.

e.g., SELECT name, job, years
FROM Staff
ORDER BY years DESC

[8] Predicates

[8.1] Basic Predicates

. Comparison operators: =, <>, <, >, <=, >=

. String operators: concatenation ||

. Boolean (logical) operators: AND, OR, NOT.

e.g., SELECT title
FROM Movie
WHERE year > 1970 AND NOT incolor

. IS NULL and IS NOT NULL to check the null values.

Cannot use = NULL or <> NULL.

e.g., SELECT DISTINCT name
FROM Product

WHERE color IS NOT NULL

. Data types must be compatible.
. Enclosing a character value in single quotation marks, e.g., WHERE job = 'clerk'.
. No quotation marks for numeric values, e.g., WHERE age = 25.
. SQL is case insensitive. But values inside quotes are case sensitive.

[8.2] LIKE Predicates

s LIKE P s
NOT LIKE P

. Compare strings, s is a string and p is a pattern.
. Use the under score (_) to represent any single character.
. Use the percent sign (%) to represent a string of zero or more characters.

e.g., SELECT title
FROM movie
WHERE title LIKE 'The%'

will select the movies whose title begins with 'The'.

e.g., SELECT title
FROM movie
WHERE title LIKE '%''s%'

will select the movies whose title includes an apostrophe, e.g., Logan's Run.

[8.3] Conditions Involving Relations

Below, s is a scalar value and R is a one-column relation.
. EXISTS R is true if and only if R is not empty.
. s in R is true if and only if s is equal to one of the values in R.
. s > ALL R is true if and only if s is greater than every value in R.
. s > ANY R is true if and only if s is greater than at least one value in R.
. Put NOT in front of entire expression to negate the EXISTS, ALL and ANY operators.
e.g., NOT EXISTS R, NOT s > ALL R, NOT s > ANY R.

[8.4] Conditions Involving Tuples

. Compare a tuple t which has the same number of components as a relation R.
t IN R, t <> ANY R

e.g., WHERE (title, year) IN (SELECT movieTitle, movieYear
FROM StarsIn
WHERE starName = 'Harrison Ford')

[9] Aggregation Operators

. Work on columns of relations rather than tuples. Always produce a single(scalar) values.
. SUM: sum of the values in the column.
. AVG: average.
. MAX: maximum.

. MIN: minimum
. COUNT: the number of values (including duplicates unless eliminated with DISTINCT).

[10] Set Operators

. UNION, INTERSECT, EXCEPT
. No duplicates in default.
. Prevent the elimination of duplicates: UNION ALL, INTERSECT ALL, EXCEPT ALL.

[11] SQL DDL

[11.1] Create Tables

CREATE TABLE <name> (<attr1> type, <attr2> type, ... <attrn> type)

CREATE TABLE <... <attr> type DEFAULT <value> ...)

[11.2] Types

. char(n): fixed length string of exactly n characters with maximum length of 254.
. varchar(n): variable length string of up to n characters with maximum length of 4000.
. smallint: signed integer (2 bytes).
. int: signed integer (4 bytes).
. real: real numbers (single-precision floating point 32 bits).
. double: real numbers (double-precision floating point 64 bits).
. decimal(n,m): real number (precision is n, scale is m).
. date: three-part value, e.g., 2003-06-17.
. time: three-part value, e.g., 18.15.05.
. timestamp: seven-part value, e.g., 2003-06-17-18.15.05.000000.
. null value: a special value that is distinct from all non-value and exits for all data types.

II. Exercise

Given the database schema:
AIRPORT(city, country, runways);
FLIGHT(flightId, day, depart_city, arrive_city, depart_time, arrive_time, plane_type);
PLANE(plane_type, capacity);

write the following queries in SQL.

a. Find airports with late flights to Toronto on a big plane. "Late" means departing after 9pm; "big" means with capacity 150 passengers or more.

```
SELECT F.depart_city
FROM FLIGHT F, PLANE P
WHERE F.plane_type = P.plane_type
  AND F.arrive_city = 'Toronto'
  AND F.depar_time > '2100'
  AND P.capacity >= 150
```

b. Find types of planes that land at airport which either have one runway, or where Boeing 747s do not land.

```
SELECT F.plane_type
FROM FLIGHT F, AIRPORT A
WHERE (F.arrive_city = A.city AND A.runways = 1)
        OR F.arrive_city NOT IN (SELECT arrive_city
                                 FROM FLIGHT
                                 WHERE plane_type = 'Boeing747')
```

c. For each type, find the minimum number of runways at an airport from which it takes off on Sundays.

```
SELECT F.plane_type, MIN(A.runways)
FROM FLIGHT F, AIRPORT A
WHERE F.depart_city = A.city
          AND F.day = 'Sunday'
GROUP BY F.plane_type
```

d. Find the number of international flights that leave Canada each Monday.

```
SELECT COUNT(*)
FROM FLIGHT F, AIRPORT A1, AIRPORT A2
WHERE F.depart_city = A1.city
    AND F.arrive_city = A2.city
    AND A1.country = 'Canada'
    AND A2.country <> 'Canada'
    AND F.day = 'Monday'
```