

Creating SQL Syntax

I. Introduction to SQL syntax

All programming language such as SQL follow a set of guidelines called *syntax*. Syntax sets the procedures on how to create the structure of SQL queries. The basic structure is the *statement*. SQL statements consist of clauses and end with a semicolon (;). Semicolons are also used to separate multiple statements. There are many types of SQL statements but since we will do more of querying and viewing of database, this session will only focus on one type which is the SELECT statement.

II. Creating SQL Syntax

SELECT statement is used to query data from different records in an existing database. The statement has the following clauses:

- SELECT – specifies the variables to be retrieved
- FROM – specifies the record to be accessed
- WHERE – specifies which observations/cases in the FROM record are to be used

The basic SELECT statement has the following syntax:

```
SELECT variables
FROM records
[WHERE (condition)];
```

The WHERE clause is optional; if not specified, then all observations are used.

A. SELECT clause

The SELECT clause is mandatory. It specifies a list of variables to be retrieved from the records in the FROM clause. It has the following general format:

```
SELECT [ALL or DISTINCT] variables
```

The ALL and DISTINCT are optional. DISTINCT specifies that duplicate observations are discarded. A duplicate observation is when each corresponding variable has the same value. If not specified, then it will retain all duplicate observations. A comma is used to separate the variables in the list. In addition, the asterisk is used to display all variables in the records specified in the FROM clause. If the variables in list are from different records then specify first the record (where the variable name is located) followed by the variable name separated by a dot(.).

Examples:

```
SELECT msname, mfname
```

```
SELECT *
```

```
SELECT hpq_mem.*, hsize
```

B. FROM clause

The FROM clause is also mandatory, and always follows the SELECT clause. It lists records accessed by the query. The general format is:

```
SELECT variables
FROM records
```

When the FROM list contains multiple records, commas are used to separate the record names. Also, when the FROM list has multiple records, they must be *joined* together. The JOIN condition will be discussed in detail later.

Examples:

```
SELECT msname, mfname  
FROM hpq_mem
```

```
SELECT *  
FROM hpq_mem
```

```
SELECT hpq_mem.*, hsize  
FROM hpq_mem, hpq_hh  
WHERE hpq_hh.hcn=hpq_mem.hcn
```

C. *WHERE* clause

The WHERE clause is optional. When specified, it always follows the FROM clause. The WHERE clause filters the observations specified from the FROM clause. When no condition is specified then all observations are used. The WHERE keyword is followed by logical expressions. The general format is:

```
SELECT variables  
FROM records  
WHERE logical expression
```

Examples:

```
SELECT msname, mfname  
FROM hpq_mem  
WHERE reln=1;
```

```
SELECT *  
FROM hpq_mem  
WHERE reln=1;
```

```
SELECT hpq_mem.*, hsize  
FROM hpq_mem, hpq_hh  
WHERE hpq_hh.hcn=hpq_mem.hcn AND reln=1;
```

D. Logical Expression

Logical expressions compare values against other values or perform arithmetic calculations. These are also used to refine search for specific observations. Logical expressions are composed of operators, and are grouped by parentheses. There are two types of operators:

1. Comparison Operators - compare the contents of a variable with the specified value in the condition. The table below shows and describes common comparison operators.

Comparison Operators	Description
=	equal to
<>, !=	is not equal to
<	less than
>	greater than
>=	greater than or equal to
<=	less than or equal to

Example:

```
SELECT msname, mfname, age_yr
FROM hpq_mem
WHERE age_yr>=10;
```

```
SELECT *
FROM hpq_mem
WHERE age_yr<5;
```

- Logical Operators- compare two conditions at a time to determine whether the observations satisfy the conditions. The table below shows and describes common logical operators:

Logical Operators	Description
AND	For an observation to be selected all the specified conditions must be true.
OR	For an observation to be selected at least one of the conditions must be true.
NOT	For an observation to be selected the specified condition must be false.

Examples:

```
SELECT msname, mfname, age_yr
FROM hpq_mem
WHERE age_yr>=6 AND age_yr<=16;
```

```
SELECT msname, mfname, age_yr
FROM hpq_mem
WHERE age_yr<=5 AND ( mnutind=3 OR mnutind=4);
```

```
SELECT *
FROM hpq_mem
WHERE sex is NOT null
```

E. Join Condition

To display information from two more records, the JOIN condition is used. The JOIN condition is applied in the WHERE clause. Since the information is being retrieved from two tables, the common variable

between the two records should be identified. This common variable will be used to merge the information from the two or more records. If the variables being displayed (in the SELECT clause) are common in two records, it should be specified from what record the variables will be retrieved. The general format is:

```
SELECT record1.variable, record2.variable  
FROM record1, record2  
WHERE record1.commonvariable=record2.commonvariable
```

Examples:

```
SELECT msname, mfname, hsize  
FROM hpq_mem, hpq_hh  
WHERE hpq_hh.hcn=hpq_mem.hcn AND reln=1;
```

```
SELECT DISTINCT hpq_mem.brgy, hpq_mem.hcn, hpq_mem.purok, hpq_mem.hcn, msname, mfname,  
hsize  
FROM hpq_hh, hpq_mem  
WHERE hpq_mem.hcn=hpq_hh.hcn and reln=1;
```