Inside Module 4

Selecting Records with Suprtool

- Using the If command: Page 2
- Comparing fields: Page 4
- Selecting by pattern-matching: Page 7
- Let’s do a crossword puzzle: Page 9
- Dates as selection criteria: Page 10
- Selecting on partial fields: Page 18
- Using tables to select records: Page 23
- Using Tables with Keyed reads: Page 32
Selecting records

- You can use the IF command to choose records by selecting ranges of numbers, dates, or multiple criteria
  
  > if sales-qty >= 100 and sales-qty < 5000
  
  > if cust-status = 10,20,30,35

- Only one IF command is permitted per task

- Suprtool uses short-circuit evaluation. e.g.

  > if age > 70 and sex = "M"

  should be faster than:

  > if sex = "M" and age > 70
More options to specify selection criteria

You can also use these words and signs to select records:

- AND, OR and NOT operators
- parentheses: ) or ( 
- relational operators: = < > >= <= <> 
- pattern matching: == and >=
Comparing fields

- You can compare one field to another
  > `if deliv-date = purch-date`

- You can compare a numeric field to a calculation
  > `if sales-total <> product-price * sales-qty`

- You can compare a field to a constant
  > `if cust-status = "OK", "DEAC"`
Arithmetic If expressions

- Select records based on arithmetic expressions

  > if unit-cost * sales-qty > 10000

  > if sales-total < sales-qty * product-price + sales-tax

- Use parentheses to keep things clear
Field types and sizes in comparisons

- Byte and character fields can be different sizes, but...
  - comparison is for length of shorter field
  - comparison ignores last bytes of longer field
Selecting records by pattern-matching

Pattern-matching

- Includes or excludes values in specified fields using these operators
  - `==` selects records that match pattern
  - `<>` selects records that do not match pattern

- Can be used only on character fields
- Can specify multiple selection criteria
- Can use special characters to define selection criteria
Special characters in pattern-matching

- Use these special characters to match patterns:
  
  @ represents any *string* of characters
  
  ? represents one *alphanumeric* character
  
  # represents one *numeric* character
  
  ~ represents zero or more *blanks*
  
  & indicates the next character is *literal*
Exercise 1  
Solve a crossword puzzle

- Use Suprtool to solve this crossword puzzle:
  - an 8 letter word
  - meaning “most befuddled or dazed”
  - second letter is an “o”
  - fourth letter is a “z”

- HINT: Suprtool has a spelling checker. Each word in its dictionary is stored as one record.
Identifying a field as a date

- First use the ITEM command to identify a field as a date:
  - `item transaction-date, date, mmddyy`
  - `item date-of-birth, date, phdate`
  - `item disbursement-date, date, ccyymmdd`

- Then use the IF command to select records:
  - `if transaction-date = $today and & date-of-birth < $date(1950/01/01) & and disbursement-date >= & $date(*+5/*/*)`
$DATE - Supported Date Formats

1. YYMMDD    MMDDYY    DDMMYY
   YYYYMMDD / CCYYMMDD  MMDDYYYY  DDMMYYYY
2. YYMM
   YYYYMM / CCYYMM  MMYYYY
3. CCYY
4. YYYYMMDD
5. AAMMDD    MMDDAA    DDMMAA
   AAMM
6. YYDDD
   CCYYDDD
7. ASK, Calendar, HPCalendar, Oracle, PHDate, SRNChronos
Dates as selection criteria

- You can select records by specifying date criteria
  
  > `item purch-date, date, phdate`
  
  > `if purch-date = $date(98/11/30) {Nov. 30, 1998}`

- You can also select a range of dates (e.g., all of December 1998)
  
  > `if purch-date > $date(98/11/30) and & purch-date < $date(99/01/01)`
  
  > `if purch-date >= $date(98/12/01) and & purch-date <= $date(98/12/31)`
Choosing records by relative date

- The $TODAY function optionally accepts an argument that indicates the number of days before or after the current day

> item expiry, date, yymmdd
> if expiry = $today {today}
> if expiry = $today (-1) {yesterday}
> if expiry > $today (+14) {more than 2 weeks away}

- Suprtool converts the $DATE function into a constant

> item date-field, date, mmddyy
> if date-field = $date(*/6/*) {six months ago}
> if date-field = 091898 {if today is Mar. 18, 1999 (constant)}
**Dates must collate correctly for > and <**

- $DATE gets converted to a constant
- For ddmmyy or mmddyy dates, the constant is in that format
- ddmmyy and mmddyy dates don't sort properly
- Suprtool rejects greater than or less than comparisons for them
- Error: Invalid date format for the comparison
- Use $STDDATE for non-collating dates
Use $STDDATE for non-collating dates

- Turn a non-collating date into CCYYMMDD format:
  >item purch-date,date,mmddyy
  >if $stddate(purch-date) < $today

- Compare dates in two different formats by converting them both to CCYYMMDD format:
  >item purch-date,date,mmddyy
  >item deliv-date,date,ddmmyyyy
  >if $stddate(purch-date) <= $stddate(deliv-date)

- Dates must be valid for $stddate to work:
  >item purch-date,date,mmddyy
  >if not invalid(purch-date) and & $stddate(purch-date) < $today
Date Arithmetic

- You can calculate the difference between 2 dates using the $days function.

- $days converts a date to the juliandays date format. I.e. the number of days since a base date (4713 BC).

  item purch-date,date,YYYYMMDD
  item deliv-date,date,YYYYMMDD
  if $days(deliv-date) - $days(purch-date) > 5

- Invalid dates return value 0 (zero)
Converting days back to dates

- **Juliandays** date format represents days offset from 4713 BC

- Combine **juliandays** with **$stddate** to convert result of **$days** calculations:
  
  ```
  >...  
  >extract latest-delivery = ($days(date-ord) + 7)  
  >xeq  
  >...  
  >item latest-delivery, date, juliandays  
  >item deliv-date, date, YYYYMMDD  
  >extract deliv-date = $stddate(latest-delivery)
  ```
Verify that dates are valid

- Use $INVALID to select records with invalid dates
  
  ```
  >item purch-date,date,yyymmdd
  >if $invalid(purch-date)
  >list standard title “Records with bad dates”
  ```

- Or use it to deselect invalid dates
  
  ```
  >if not $invalid(purch-date) and &
  purch-date > $date(*/*-6/*)
  ```
Year 2000 dates

- Some selections generate “invalid” date constants, if the date field cannot hold century information and the constant would be in the next century
  >item purch-date,date,yyymmdd
  >if purch-date > $date(*+5/**/*)
  Error: Cannot use a date beyond 1999 for this format

- You can override this error condition
  >set date ifyy2000error off

- Or you can use $STDDATE to assume a century
  >set date cutoff 50
  >if $stddate(purch-date) > $date(*+5/**/*)
$\text{truncate, Mod mod and abs functions}$

- $\text{truncate}$ returns “whole number”, i.e. drops decimals
  $\text{truncate}(127.2 / 12) = 10$

- $\text{Mod}$ returns the remainder
  $7 \mod 5 = 2$

- $\text{abs}$ returns the absolute value (no sign)
  $\text{abs}(-121) = 121$
Selecting on parts of a number

- You can select any part of a numeric field with the If command

- Use a divide operation to select on the high-order digits
  \[ \text{if } \text{truncate}(\text{ord-date-yyymmdd} / 100) = 9812 \]

- Use MOD to select on the low-order digits
  \[ \text{if } \text{ord-date-yyymmdd} \mod 100 \leq 15 \]

- Use divide and MOD together to select on middle digits
  \[ \text{if } (\text{truncate}(\text{ord-date-yyymmdd} / 100) \mod 100) \leq 02 \]
Calculating day of week

- Juliandays measures offset from a Monday

- Combine $days$ with mod to calculate day-of-week
  
  ```
  >ite ord=date,date,YYYYMMDD
  >ext day = ($days(dt) mod 7)
  ```

  0 = Monday
  1 = Tuesday
  2 = Wednesday
  ....
  6 = Sunday
Comparing sub-fields

- You can select any part of a character field with the IF command.

- If we define a street-address field as 2X25, any part of this field can be selected:
  > if street-address(2) = "Canada"
  > if street-address(1,7,2) = "10"
  > if street-address(1,13) = "Marine Drive"
Testing byte type fields

- You can test if a byte type field contains alpha, numeric, alphanumeric or special characters

  >if cust-account = numeric

  >if street-address <> alphanumeric

- You can also check for an ASCII character by specifying its numeric value or control letter

  >define any-char,1,1,byte

  >if any-char = ^13 {if byte is a Return}

  >if any-char = ^G {if byte is a Bell}
Checking bits within a field

- The IF command can select records based on bit values in a field
  \[
  \text{if cust-status.(3:1) = 1} \\
  \text{if cust-status.(3:2) = 0}
  \]

- Bit checking only works for 16-bit words

- Field must be \textit{Integer} or \textit{Logical}
Extending the If command

- You can extend the length of an IF command beyond the 256 character limit by using the $READ function.

```plaintext
>get m-customer
>if $read
-name-last == "@Kirk@" and
-state-code = "BC"
-and
-cust-account > 12
-//
```

- $READ prompts for the next line of the IF expression until it encounters a Return or a double slash (//).
Creating tables as selection criteria

- The TABLE command creates a set of values that can be used as selection criteria:

  `TABLE tablename, itemname, table-keyword, table-values`

  ```
  >table select,transcode,item,"BUY","SELL"
  >table cust-table,cust-num,file,custfile
  ```

- The source of input can be an item value or a file

- The TABLE command sorts values as they are loaded into a table
Table characteristics

- Only *one key* can be put into a table
- Suprtool can handle up to *ten tables*
- Each table can have up to *two gigabytes* of data on MPE
- 500 Mbs in total on HP-UX
- Tables are *temporary* structures that are reset when a task has been completed
- You can *hold* a table so it is not reset
- Table values are *sorted*
When would I use a table?

- Instead of listing all the values
  
  ```
  if field = value1, value2, value3
  ```

- When there are too many values to fit in an IF command

- When the selection values change occasionally

- When the selection is based on the results of a prior task
Loading a table with values from a file

- If the file containing the values is not sorted, specify FILE as the keyword
  > table states,st-code,file,western.data
  > if qty-ship < qty-order and $lookup(states,st-code)

- If the file is sorted, specify SORTED as the keyword
  > table states,st-code,sorted,western.data
  > if qty-ship < qty-order and $lookup(states,st-code)

- The field selected from the input file must have exactly the same format as the table
How does the Table command find a field?

- If the input file is self-describing, Suprtool finds the location of the field via the user label.

- If the file is not self-describing, or the named field is not found in the file label, Suprtool loads the requested data from the start of each record.
Inserting items into a table

- You can also use the TABLE command to insert hardcoded values
  - Specify ITEM as the table keyword
  - `table states, st-code, item, "WA", "OR", "CA"
  - `table states, st-code, item, "WI", "ID", "NE"
  - `table states, st-code, item, "NM", "AK", "HI"
  - `if cust-status = "OK" and $lookup(states, st-code)
Selecting input records that match a value in a table

- Use the $LOOKUP function with the IF command to select records that match a value in a table
  >if $lookup(cust-table,cust-acct)

- If the $LOOKUP function finds a match, the expression is true

- If there are multiple conditions in the IF expression, the expression is evaluated faster when $LOOKUP is the last condition
  >if status = "10" and $lookup(cust-table,cust-acct)

- Use NOT to select records which don’t match table values
Saving and deleting tables

- The HOLD option tells Suprtool to save a table after a task has been completed
  
  > `table states, st-code, file, western.data`
  
  > `table parts, part-no, file, partin, hold`

- The RESET TABLE command clears all the tables. You cannot reset individual tables.
  
  > `reset table`
Can we find all the invoices for BC customers and sort them by customer ID?

- The invoice records are in the sales detail dataset, but state-code is in the customer master record

```plaintext
> get m-customer
> if state-code = "BC"
> extract cust-account
> output bccust
> xeq
> table bc,cust-account,file,bccust
> get d-sales
> if $lookup(bc,cust-account)
> sort cust-account
> list standard
> xeq
```
Selecting records using the Chain command

- Alternately, you can use the CHAIN command to find the required invoices after you have created an output file of British Columbia customers (Bccust)

  
  ```
  >table brit,cust-account,file,bccust
  >chain d-sales,cust-account=brit
  >list standard
  >xeq
  ```

- The CHAIN command performs keyed retrievals for the values in the table.

- No SORT command is necessary because the CHAIN command retrieves the records in the same order as they are found in the table.
String Functions and Features

- $TRIM,$RTRIM,$LTRIM
- $UPPER,$LOWER
- + Operator and Target field
Summary

- IF command
- Field comparison
- IF expressions (Boolean operators, parentheses)
- Pattern-matching
- Date fields
- Sub-field comparisons
- $READ function
- Tables
- Selecting from one file based on criteria in another file