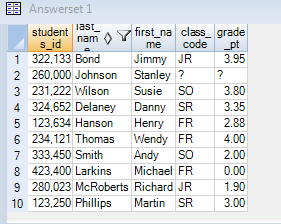
**TERADATA ASSIGNMENTS**

**LAB 5**

***THE WHERE CLAUSE***

**SELECT** \* **FROM** Student\_Table**;**

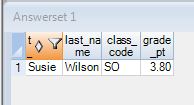


***LIMITS RETURNING ROWS***

**SELECT** FIRST\_NAME,LAST\_NAME,CLASS\_CODE,GRADE\_PT

**FROM** STUDENT\_TABLE

**WHERE** FIRST\_NAME='SUSIE'**;**

****

***USING COLUMN ALIAS THROUGHOUT SQL***

**SEL** FIRST\_NAME **AS** FNAME

,LAST\_NAME LNAME

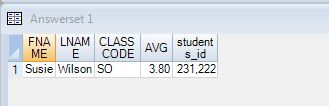
,CLASS\_CODE "CLASS CODE"

,GRADE\_PT **AS**"AVG"

,STUDENTS\_ID

**FROM** STUDENT\_TABLE

**WHERE** FNAME='SUSIE'**;**

****

***DOUBLE QUOTED ALIASES ARE RESERVED FOR SPACES***

**SEL** FIRST\_NAME **AS** FNAME

,LAST\_NAME LNAME

,CLASS\_CODE "CLASS CODE"

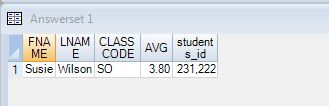
,GRADE\_PT **AS**"AVG"

,STUDENTS\_ID

**FROM** STUDENT\_TABLE

**WHERE** FNAME='SUSIE'

**ORDER** **BY** "AVG"**;**

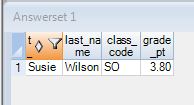


***CHARACTER DATA NEEDS SINGLE QUOTES IN THE WHERE CLAUSE***

**SELECT** FIRST\_NAME,LAST\_NAME,CLASS\_CODE,GRADE\_PT

**FROM** STUDENT\_TABLE

**WHERE** FIRST\_NAME='SUSIE'**;**

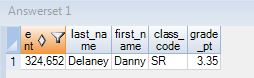
****

***CHARACTER DATA NEEDS SINGLE QUOTES BUT NUMBER DON’T***

**SEL** \*

**FROM** STUDENT\_TABLE

**WHERE** GRADE\_PT=3.35**;**



***NULL MEANS UNKNOWN DATA SO EQUAL (=) WON’T WORK***

**SEL** \*

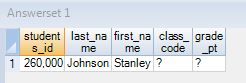
**FROM** STUDENT\_TABLE

**WHERE** CLASS\_CODE=**NULL;**

**SEL** \*

**FROM** STUDENT\_TABLE

**WHERE** CLASS\_CODE **IS** **NULL;**



***NULL IS UNKNOWN DATA SO NOT EQUAL WON’T WORK***

**SEL** \*

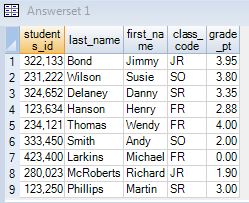
**FROM** STUDENT\_TABLE

**WHERE** CLASS\_CODE = **NOT** **NULL;**

**SEL** \*

**FROM** STUDENT\_TABLE

**WHERE** CLASS\_CODE **IS** **NOT** **NULL;**

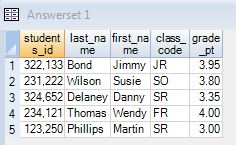


***USING GREATER THAN OR EQUAL TO OPERATOR (>=)***

**SEL** \*

**FROM** STUDENT\_TABLE

**WHERE** GRADE\_PT >=3.0**;**

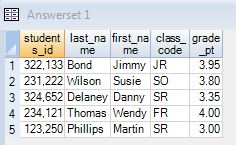


***USING GE AS GREATER THAN OR EQUAL TO (>=)***

**SEL** \*

**FROM** STUDENT\_TABLE

**WHERE** GRADE\_PT GE 3.0**;**



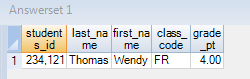
***AND IN THE WHERE***

**SEL** \*

**FROM** STUDENT\_TABLE

**WHERE** CLASS\_CODE = 'FR'

**AND** FIRST\_NAME='WENDY'**;**



***TROUBLESHOOTING AND***

**SEL** \*

**FROM** STUDENT\_TABLE

**WHERE** GRADE\_PT =3.0 **AND** GRADE\_PT=4.0**;**

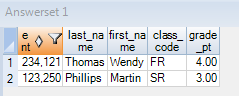
**NO ROWS RETURNED**

***OR IN THE WHERE***

**SEL** \*

**FROM** STUDENT\_TABLE

**WHERE** GRADE\_PT =3.0 **OR** GRADE\_PT=4.0**;**



***TROUBLESHOOTING OR***

**SEL** \*

**FROM** STUDENT\_TABLE

**WHERE** GRADE\_PT =3.0 **OR** 4.0**;**

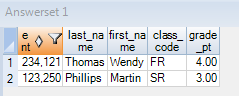
**ERROR OCCURS**

***OR MUST UTILISE SAME COLUMN NAMES***

**SEL** \*

**FROM** STUDENT\_TABLE

**WHERE** GRADE\_PT =3.0 **OR** GRADE\_PT=4.0**;**



***TROUBLESHOOTING CHARACTER DATA***

**SEL** \*

**FROM** STUDENT\_TABLE

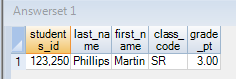
**WHERE** GRADE\_PT =3.0 **AND** CLASS\_CODE=SR**;**

**ERROR OCCURS**

**SEL** \*

**FROM** STUDENT\_TABLE

**WHERE** GRADE\_PT =3.0 **AND** CLASS\_CODE='SR'**;**



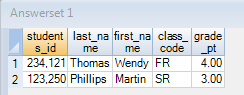
***QUIZ – HOW MANY WILL RETURN?***

**SEL** \*

**FROM** STUDENT\_TABLE

**WHERE** GRADE\_PT =3.0 **OR** GRADE\_PT=4.0**;**

**AND** CLASS\_CODE ='SR'**;**



**ANSWER IS 2 ROWS**

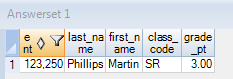
***USING PARANTHESIS TO CHANGE THE ORDER OF PRECEDENCE***

**SEL** \*

**FROM** STUDENT\_TABLE

**WHERE** (GRADE\_PT =3.0 **OR** GRADE\_PT=4.0)

**AND** CLASS\_CODE ='SR'**;**



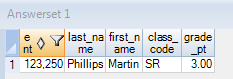
***Using an IN LIST in place of OR***

**SEL** \*

**FROM** STUDENT\_TABLE

**WHERE** GRADE\_PT **IN** (3.0 ,4.0)

**AND** CLASS\_CODE ='SR'**;**

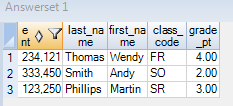


***The IN LIST is an excellent technique***

**SEL** \*

**FROM** STUDENT\_TABLE

**WHERE** GRADE\_PT **IN** (2.0,3.0 ,4.0)**;**



***IN list vs OR brings the same results***

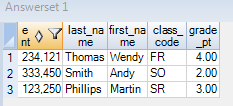
**SEL** \*

**FROM** STUDENT\_TABLE

**WHERE** GRADE\_PT = 2.0

**OR** GRADE\_PT = 3.0

**OR** GRADE\_PT = 4.0**;**

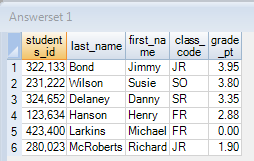


***Using a NOT IN LIST***

**SEL** \*

**FROM** STUDENT\_TABLE

**WHERE** GRADE\_PT **NOT** **IN** (2.0,3.0 ,4.0)**;**



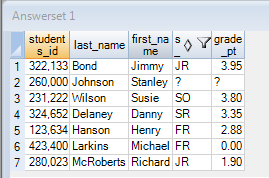
***TECHNIQUE FOR HANDLING NULL IN NOT IN LIST***

**SEL** \*

**FROM** STUDENT\_TABLE

**WHERE** GRADE\_PT **NOT** **IN** (2.0,3.0 ,4.0)

**OR** GRADE\_PT **IS** **NULL;**

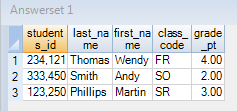


***AN IN LIST WITH KEYWORD ANY***

**SEL** \*

**FROM** STUDENT\_TABLE

**WHERE** GRADE\_PT = ANY (2.0,3.0 ,4.0)**;**

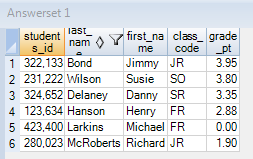


***A NOT IN LIST WITH KEYWORDS NOT=ALL***

**SEL** \*

**FROM** STUDENT\_TABLE

**WHERE** GRADE\_PT **NOT**= **ALL** (2.0,3.0 ,4.0)**;**

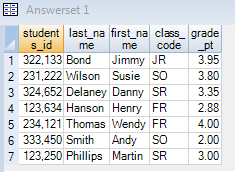


***BETWEEN is inclusive***

**SEL** \*

**FROM** STUDENT\_TABLE

**WHERE** GRADE\_PT between 2.0 **and** 4.0**;**

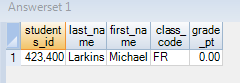


***BETWEEN works for character data***

**SEL** \*

**FROM** STUDENT\_TABLE

**WHERE** last\_name between 'L' **and** 'LZ'**;**

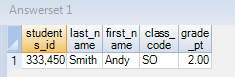


***LIKE USES WILDCARDS PERCENT ‘%’ AND UNDERSCORE ‘\_’***

**SEL** \*

**FROM** STUDENT\_TABLE

**WHERE** last\_name LIKE 'SM%'**;**

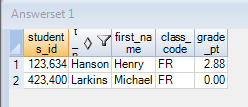


***LIKE COMMAND UNDERSCORE IS WILDCARD FOR ONE CHARACTER***

**SEL** \*

**FROM** STUDENT\_TABLE

**WHERE** last\_name LIKE '\_a%'**;**

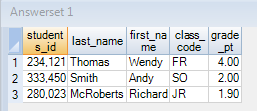


***LIKE ALL means ALL conditions must be met***

**SEL** \*

**FROM** STUDENT\_TABLE

**WHERE** last\_name LIKE **ALL** ('%M%', '%S%')**;**

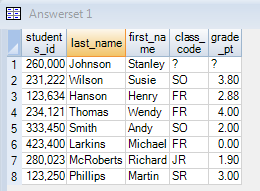


***LIKE ANY means ANY of the conditions must be met***

**SEL** \*

**FROM** STUDENT\_TABLE

**WHERE** last\_name LIKE ANY('%M%', '%S%')**;**



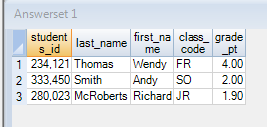
***IN ANSI TRANSACTION MODE CASE MATTERS***

***IN TREADATA TRANSACTION MODE CASE DOES NOT MATTERS***

**SEL** \*

**FROM** STUDENT\_TABLE

**WHERE** last\_name LIKE **ALL**('%S%', '%m%')**;**

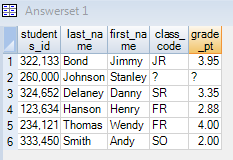


***LIKE COMMAND WORKS DIFFERENTLY ON CHAR Vs VARCHAR***

**SEL** \*

**FROM** STUDENT\_TABLE

**WHERE** first\_name LIKE '%y'**;**



***TROUBLESHOOTING LIKE COMMAND ON CHAR DATA***

**SEL** \*

**FROM** STUDENT\_TABLE

**WHERE** last\_name LIKE '%a'**;**

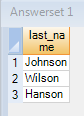
**NO ROWS RETURNED**

***TRIM COMMAND***

**SEL** last\_name

**FROM** STUDENT\_TABLE

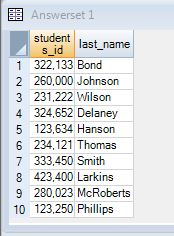
**WHERE** TRIM (last\_name) LIKE '%n'**;**



***AN EXAMPLE OF DATA WITH LEFT JUSTIFICATION AND RIGHT JUSTIFICATION***

**SEL** STUDENTS\_ID, LAST\_NAME

**FROM** STUDENT\_TABLE**;**



***ESCAPE CHARACTER IN THE LIKE COMMAND CHANGES WILDACRDS***

**SEL** \*

**FROM** STUDENT\_TABLE

**WHERE** FIRST\_NAME LIKE 'S@%' **ESCAPE** '@'**;**

***TO FIND THE WILDCARD***

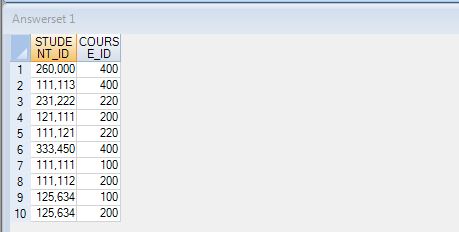
**SEL** \*

**FROM** STUDENT\_TABLE

**WHERE** TRIM(LAST\_NAME) LIKE 'T@\_' **ESCAPE** '@'**;**

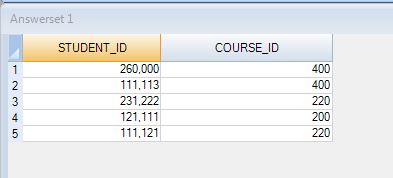
**Sample**

**SELECT** \* **FROM** STUDENT\_COURSE\_TABLE



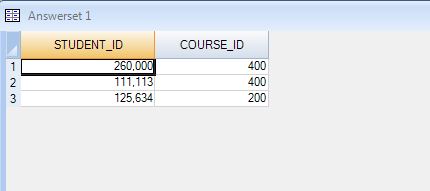
**SELECT** \* **FROM** STUDENT\_COURSE\_TABLE

**SAMPLE** 5**;**



**SELECT** \* **FROM** STUDENT\_COURSE\_TABLE

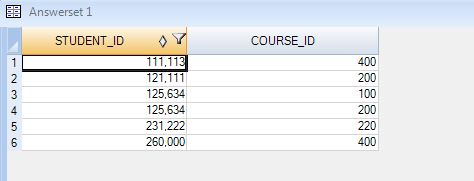
**SAMPLE** .25 **;**



**SELECT** \* **FROM** STUDENT\_COURSE\_TABLE

**SAMPLE** .25, .25

**ORDER** **BY** 1,2 **;**

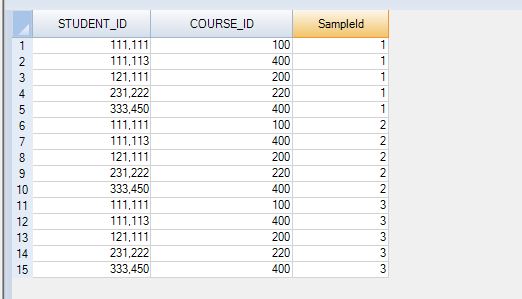


Sample with replacement

**SELECT** student\_id, course\_id,**SAMPLEID** **FROM** STUDENT\_COURSE\_TABLE

**SAMPLE** **WITH** **REPLACEMENT** 5,5,5

**ORDER** **BY** 3,1,2**;**

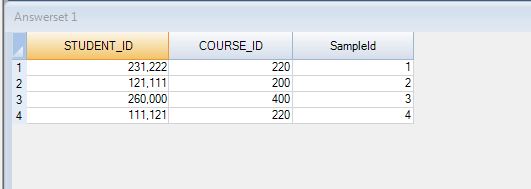


Sample with 10% sampled

**SELECT** student\_id, course\_id,**SAMPLEID** **FROM** STUDENT\_COURSE\_TABLE

**SAMPLE** .1,.1,.1,.1

**ORDER** **BY** **SAMPLEID** **;**



Sample with conditional logic

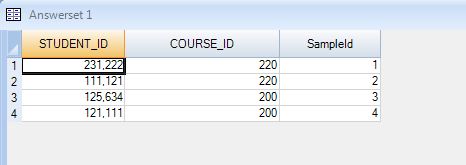
**SELECT** student\_id, course\_id,**SAMPLEID** **FROM** STUDENT\_COURSE\_TABLE

**SAMPLE** **RANDOMIZED** **ALLOCATION**

**WHEN** COURSE\_ID>200 **THEN** .1,.1 **ELSE** .2,.2

**END**

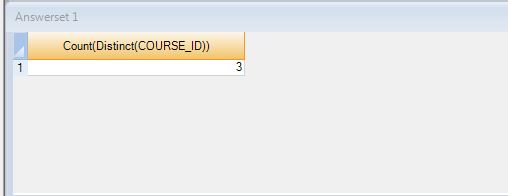
**ORDER** **BY** 3**;**



AGGREGATES and SAMPLE USING DERIVED TABLE

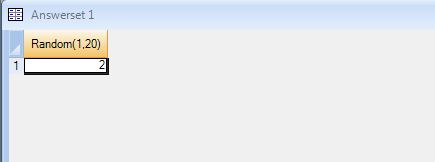
**SELECT COUNT(DISTINCT( course\_id)) FROM (SEL COURSE\_ID FROM STUDENT\_COURSE\_TABLE**

**SAMPLE 5) DT;**

****

RANDOM NUMBER GENERATOR

**SELECT** RANDOM(1,20)



RANDOM TO SELECT PERCENTAGE OF ROWS

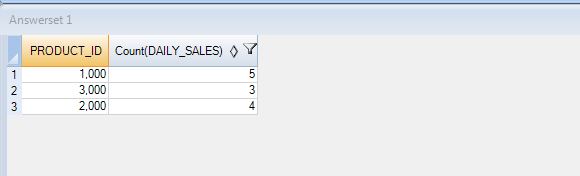
**SELECT** \* **FROM** SALES\_TABLE

**WHERE** RANDOM(1,100)=5**;**



**SELECT** PRODUCT\_ID, COUNT(DAILY\_SALES) **FROM** SALES\_TABLE

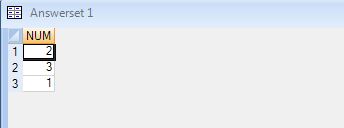
**GROUP** **BY** 1**;**

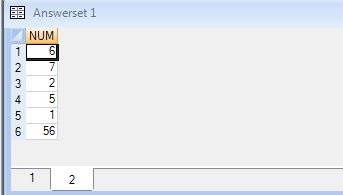


**SET OPERATORS FUNCTIONS**

**SELECT** \* **FROM** TABLE\_RED**;**

**SELECT** \* **FROM** TABLE\_BLUE**;**



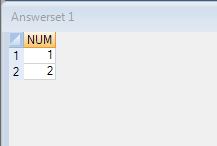


INTERSECT

**SELECT** \* **FROM** TABLE\_RED**;**

**INTERSECT**

**SELECT** \* **FROM** TABLE\_BLUE**;**

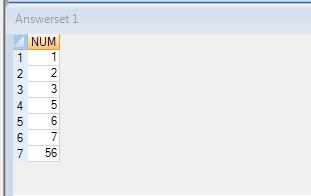
****

UNION

**SELECT** \* **FROM** TABLE\_RED

**UNION**

**SELECT** \* **FROM** TABLE\_BLUE**;**



UNION ALL

**SELECT** \* **FROM** TABLE\_RED

**UNION** **ALL**

**SELECT** \* **FROM** TABLE\_BLUE**;**

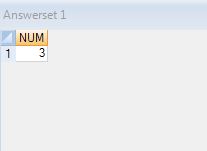


MINUS

**SELECT** \* **FROM** TABLE\_RED

**MINUS**

**SELECT** \* **FROM** TABLE\_BLUE**;**



TESTING THE KNOWLEDGE:

1)

**SELECT** \* **FROM** TABLE\_RED

**EXCEPT**

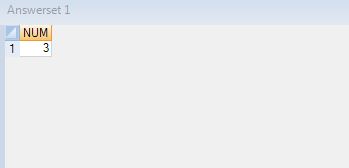
**SELECT** \* **FROM** TABLE\_BLUE**;**



**SELECT** \* **FROM** TABLE\_RED

**MINUS**

**SELECT** \* **FROM** TABLE\_BLUE**;**

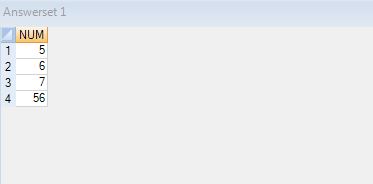


2)20

**SELECT** \* **FROM** TABLE\_BLUE

**MINUS**

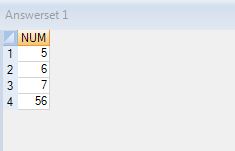
**SELECT** \* **FROM** TABLE\_RED**;**



**SELECT** \* **FROM** TABLE\_BLUE

**EXCEPT**

**SELECT** \* **FROM** TABLE\_RED**;**

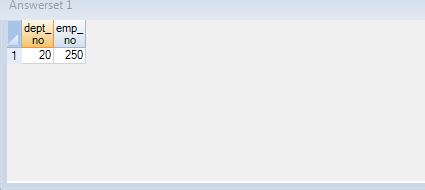


SELECT LIST

**SELECT** DEPT\_NO, EMP\_NO **FROM** EMPLOYEE\_TABLE

**INTERSECT**

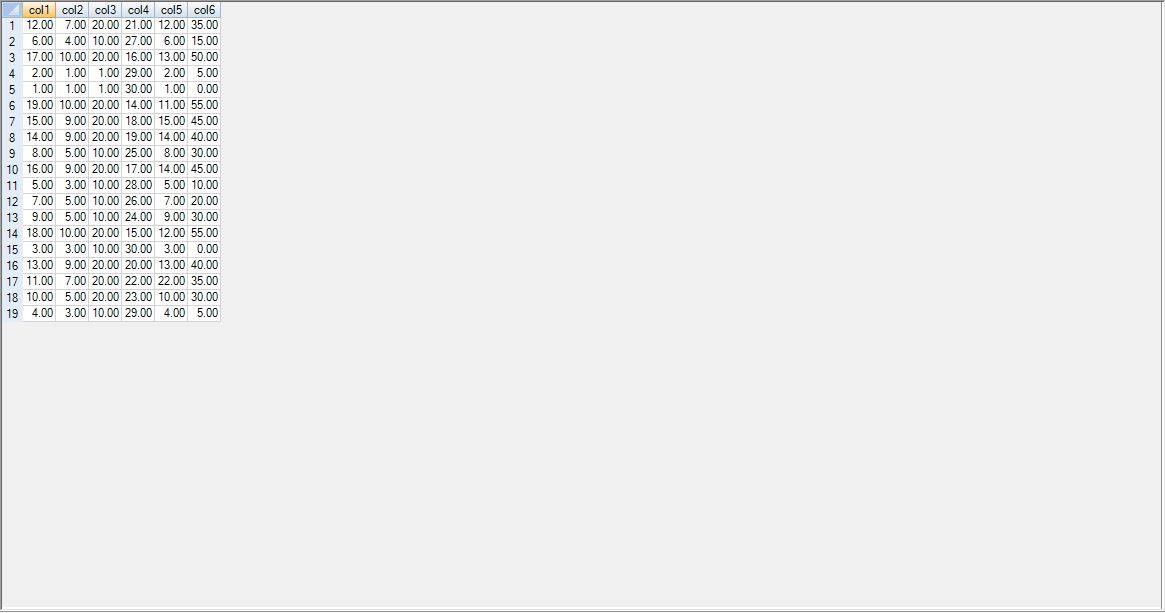
**SELECT** DEPT\_NO,MGR\_NO **FROM** DEP\_TABLE**;**

****

**STATISTICAL AGGREGATE FUNCTIONS:**

**The Stats Table:**

**SELECT** \* **FROM** Stats\_Table**;**

****

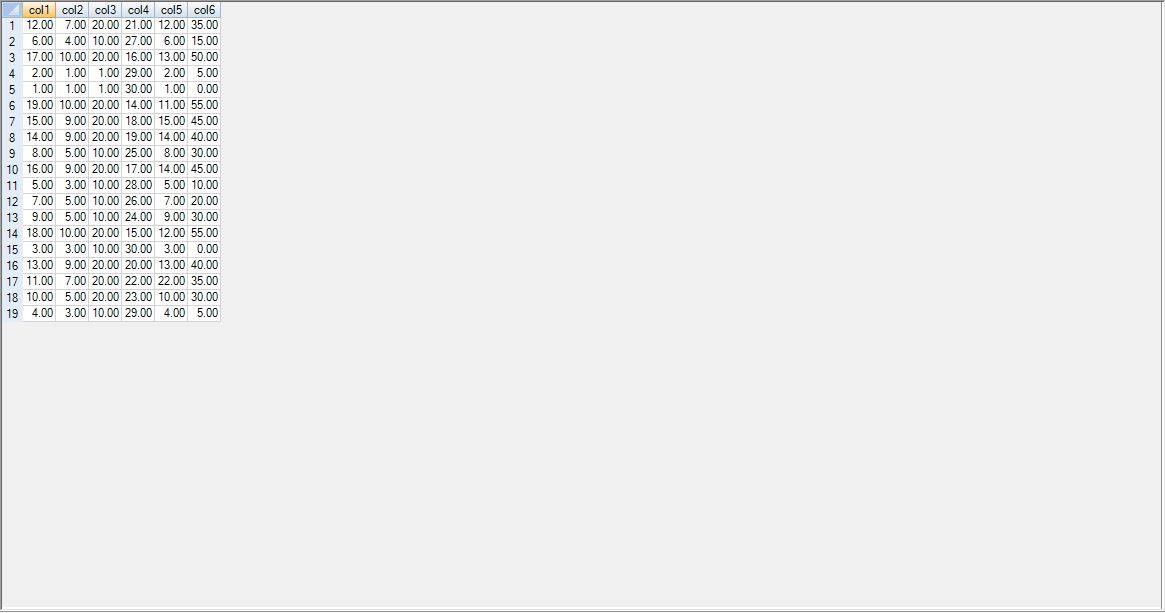
**THE KURTOSIS FUNCTION:**

**SELECT** KURTOSIS(col1) **AS** KofCol1

**FROM** Stats\_Table**;**

****

**SELECT** \* **FROM** Stats\_Table**;**

****

**SELECT** KURTOSIS(col1) **AS** KofCol1

,KURTOSIS(col2) **AS** KofCol2

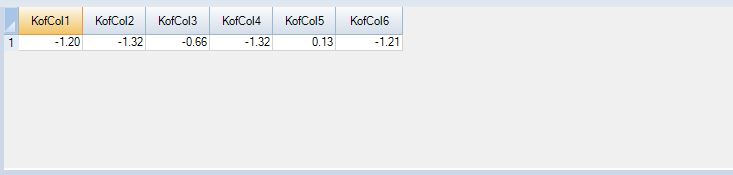
,KURTOSIS(col3) **AS** KofCol3

,KURTOSIS(col4) **AS** KofCol4

,KURTOSIS(col5) **AS** KofCol5

,KURTOSIS(col6) **AS** KofCol6

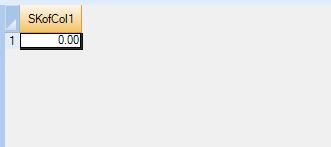
**FROM** Stats\_Table**;**



**SKEW FUNCTION:**

**SELECT** SKEW(col1) **AS** SKofCol1

**FROM** Stats\_Table**;**



**SELECT** SKEW(col1) **AS** SKofCol1

,SKEW(col2) **AS** SKofCol2

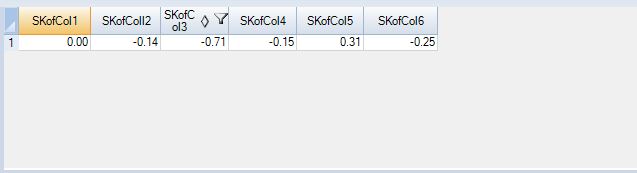
,SKEW(col3) **AS** SKofCol3

,SKEW(col4) **AS** SKofCol4

,SKEW(col5) **AS** SKofCol5

,SKEW(col6) **AS** SKofCol6

**FROM** Stats\_Table**;**



**THE STDDEV\_POP FUNCTION:**

**SELECT** STDDEV\_POP(col1) **AS** SDPCol1

**FROM** Stats\_Table**;**

****

**A STDDEV\_POP EXAMPLE:**

**SELECT** STDDEV\_POP(col1) **AS** SDPCol1

,STDDEV\_POP(col2) **AS** SDPColl2

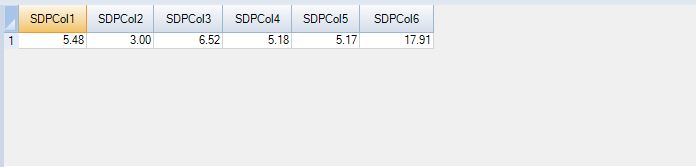
,STDDEV\_POP(col3) **AS** SDPCol3

,STDDEV\_POP(col4) **AS** SDPCol4

,STDDEV\_POP(col5) **AS** SDPCol5

,STDDEV\_POP(col6) **AS** SDPCol6

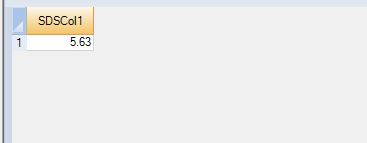
**FROM** Stats\_Table**;**

****

**THE STDDEV\_SAMP FUNCTION:**

**SELECT** STDDEV\_SAMP(col1) **AS** SDSCol1

**FROM** Stats\_Table**;**

****

**A STDDEV\_SAMP EXAMPLE:**

**SELECT** STDDEV\_SAMP(col1) **AS** SDSCol1

,STDDEV\_SAMP(col2) **AS** SDSColl2

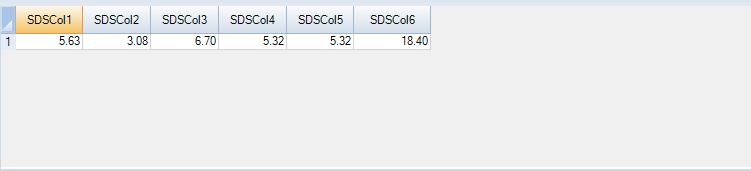
,STDDEV\_SAMP(col3) **AS** SDSCol3

,STDDEV\_SAMP(col4) **AS** SDSCol4

,STDDEV\_SAMP(col5) **AS** SDSCol5

,STDDEV\_SAMP(col6) **AS** SDSCol6

**FROM** Stats\_Table**;**



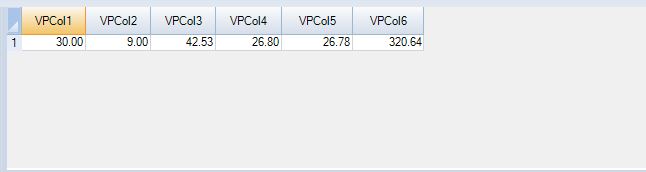
**THE VAR\_POP FUNCTION:**

**SELECT** VAR\_POP(col1) **AS** VPCol1

**FROM** Stats\_Table**;**

****

**A VAR\_POP EXAMPLE:**

****

**THE VAR\_SAMP FUNCTION:**

**SELECT** VAR\_SAMP(col1) **AS** VSCol1

**FROM** Stats\_Table**;**

****

**A VAR\_SAMP EXAMPLE:**

**SELECT** VAR\_SAMP(col1) **AS** VSCol1

,VAR\_SAMP(col2) **AS** VSCol2

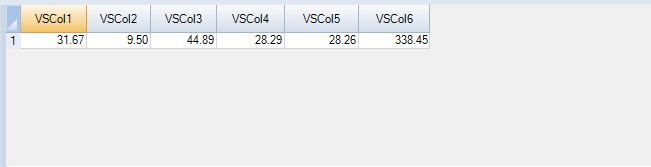
,VAR\_SAMP(col3) **AS** VSCol3

,VAR\_SAMP(col4) **AS** VSCol4

,VAR\_SAMP(col5) **AS** VSCol5

,VAR\_SAMP(col6) **AS** VSCol6

**FROM** Stats\_Table**;**

****

**THE CORR FUNCTION:**

**SELECT** CORR(col1, col2) **AS** CCol1#2

**FROM** Stats\_Table**;**

****

**A CORR EXAMPLE:**

**SELECT** CORR(col1, col2) **AS** CCol1#2

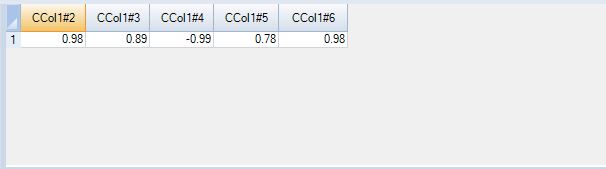
,CORR(col1, col3) **AS** CCol1#3

,CORR(col1, col4) **AS** CCol1#4

,CORR(col1, col5) **AS** CCol1#5

,CORR(col1, col6) **AS** CCol1#6

**FROM** Stats\_Table**;**

****

**ANOTHER CORR EXAMPLE:**

**SELECT** CORR(col1, col2) **AS** CCol1#2

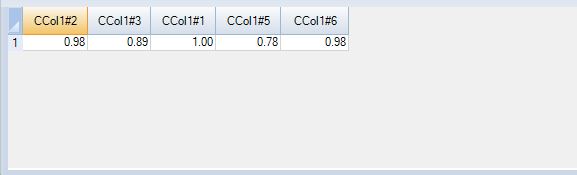
,CORR(col1, col3) **AS** CCol1#3

,CORR(col1, col1) **AS** CCol1#1

,CORR(col1, col5) **AS** CCol1#5

,CORR(col1, col6) **AS** CCol1#6

**FROM** Stats\_Table**;**

****

**THE COVAR\_POP FUNCTION:**

**SELECT** COVAR\_POP(col1, col2) **AS** CCol1#2

**FROM** Stats\_Table**;**

****

**A COVAR\_POP EXAMPLE:**

**SELECT** COVAR\_POP(col1, col2) **AS** CPCol1#2

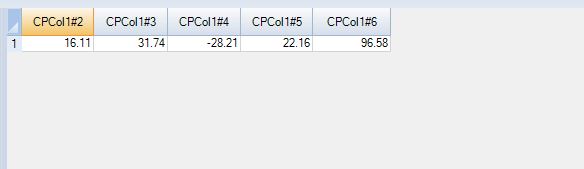
,COVAR\_POP(col1, col3) **AS** CPCol1#3

,COVAR\_POP(col1, col4) **AS** CPCol1#4

,COVAR\_POP(col1, col5) **AS** CPCol1#5

,COVAR\_POP(col1, col6) **AS** CPCol1#6

**FROM** Stats\_Table**;**

****

**ANOTHER COVAR\_POP EXAMPLE:**

**SELECT** COVAR\_POP(col1, col2) **AS** CPCol1#2

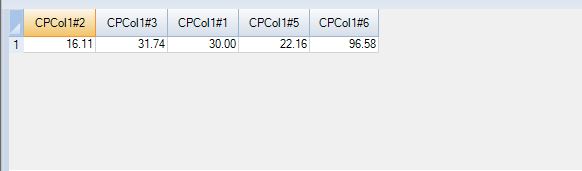
,COVAR\_POP(col1, col3) **AS** CPCol1#3

,COVAR\_POP(col1, col1) **AS** CPCol1#1

,COVAR\_POP(col1, col5) **AS** CPCol1#5

,COVAR\_POP(col1, col6) **AS** CPCol1#6

**FROM** Stats\_Table**;**



**THE REGR\_INTERCEPT FUNCTION:**

**SELECT** REGR\_INTERCEPT(col1, col2) **AS** RTofCol1#2

**FROM** Stats\_Table**;**



**A REGR\_INTERCEPT EXAMPLE:**

**SELECT** REGR\_INTERCEPT(col1, col2) **AS** RTofCol1#2

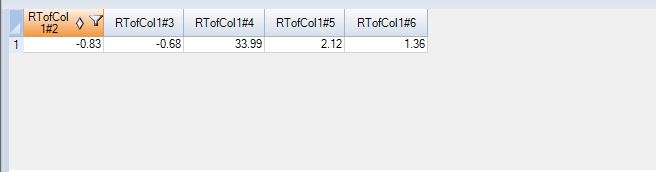
,REGR\_INTERCEPT(col1, col3) **AS** RTofCol1#3

,REGR\_INTERCEPT(col1, col4) **AS** RTofCol1#4

,REGR\_INTERCEPT(col1, col5) **AS** RTofCol1#5

,REGR\_INTERCEPT(col1, col6) **AS** RTofCol1#6

**FROM** Stats\_Table**;**

****

**ANOTHER REGR\_INTERCEPT EXAMPLE:**

**SELECT** REGR\_INTERCEPT(col1, col2) **AS** RTofCol1#2

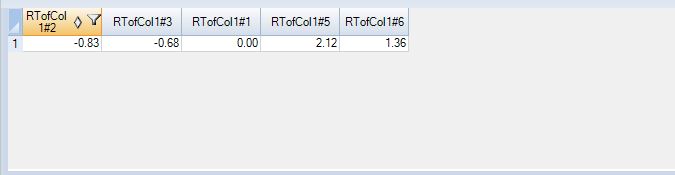
,REGR\_INTERCEPT(col1, col3) **AS** RTofCol1#3

,REGR\_INTERCEPT(col1, col1) **AS** RTofCol1#1

,REGR\_INTERCEPT(col1, col5) **AS** RTofCol1#5

,REGR\_INTERCEPT(col1, col6) **AS** RTofCol1#6

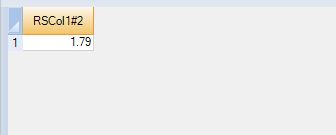
**FROM** Stats\_Table**;**

****

**THE REGR\_SLOPE FUNCTION:**

**SELECT** REGR\_SLOPE(col1, col2) **AS** RSCol1#2

**FROM** Stats\_Table**;**

****

**A REGR\_SLOPE EXAMPLE:**

**SELECT** REGR\_SLOPE(col1, col2) **AS** RSCol1#2

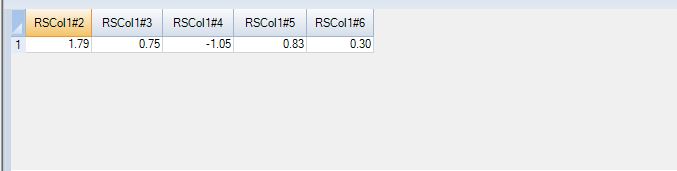
,REGR\_SLOPE(col1, col3) **AS** RSCol1#3

,REGR\_SLOPE(col1, col4) **AS** RSCol1#4

,REGR\_SLOPE(col1, col5) **AS** RSCol1#5

,REGR\_SLOPE(col1, col6) **AS** RSCol1#6

**FROM** Stats\_Table**;**

****

**ANOTHER REGR\_SLOPE EXAMPLE:**

**SELECT** REGR\_SLOPE(col1, col2) **AS** RSCol1#2

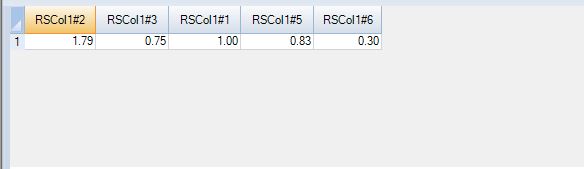
,REGR\_SLOPE(col1, col3) **AS** RSCol1#3

,REGR\_SLOPE(col1, col1) **AS** RSCol1#1

,REGR\_SLOPE(col1, col5) **AS** RSCol1#5

,REGR\_SLOPE(col1, col6) **AS** RSCol1#6

**FROM** Stats\_Table**;**

****

**USING GROUP BY:**

**SELECT** col3

,count(\*) **AS** Cnt

,avg(col1) AVG1

,stddev\_pop(col1) **AS** SD1

,var\_pop(col1) **AS** VP1

,avg(col4) **AS** AVG4

,stddev\_pop(col4) **AS** SD4

,var\_pop(col4) **AS** VP4

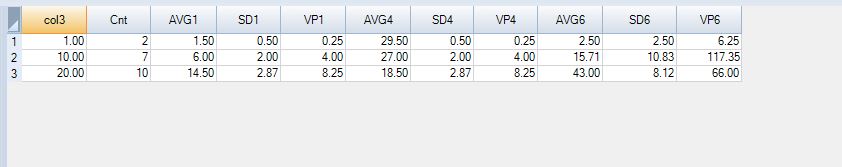
,avg(col6) **AS** AVG6

,stddev\_pop(col6) **AS** SD6

,var\_pop(col6) **AS** VP6

**FROM** Stats\_Table

**GROUP** **BY** 1 **ORDER** **BY** 1**;**



**NO HAVING CLAUSE VS USE OF HAVING CLAUSE:**

**SELECT** col3

,count(\*) **AS** Cnt

,avg(col1) AVG1

,stddev\_pop(col1) **AS** SD1

,var\_pop(col1) **AS** VP1

,avg(col4) **AS** AVG4

,stddev\_pop(col4) **AS** SD4

,var\_pop(col4) **AS** VP4

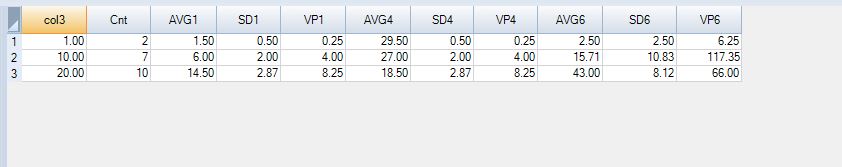
,avg(col6) **AS** AVG6

,stddev\_pop(col6) **AS** SD6

,var\_pop(col6) **AS** VP6

**FROM** Stats\_Table

**GROUP** **BY** 1 **ORDER** **BY** 1**;**



**SELECT** col3

,count(\*) **AS** Cnt

,avg(col1) AVG1

,stddev\_pop(col1) **AS** SD1

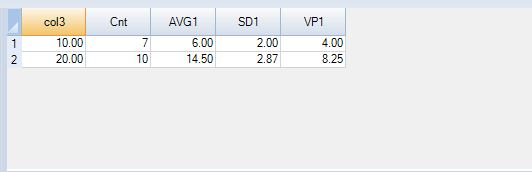
,var\_pop(col1) **AS** VP1

**FROM** Stats\_Table

**GROUP** **BY** 1

**ORDER** **BY** 1

**HAVING** Cnt>2 **AND** VP1<20**;**

****

**STORED PROCEDURE FUNCTIONS:**

**STORED PROCEDURE FUNCTIONS:**

**CREATING A STORED PROCEDURE:**

**CREATE** **PROCEDURE** First\_Procedure1()

**BEGIN**

**INSERT** **INTO** Customer\_Table **DEFAULT** **VALUES;**

**END;**

**HOW TO CALL A STORED PROCEDURE:**

**CREATE** **PROCEDURE** First\_Procedure1()

**BEGIN**

**INSERT** **INTO** Customer\_Table **DEFAULT** **VALUES;**

**END;**

**CALL** First\_Procedure1()**;**

**SELECT** \* **FROM** CUSTOMER\_TABLE**;**

****

**LABEL ALL BEGIN AND END STATEMENTS EXCEPT FIRST ONES:**

**CREATE** **PROCEDURE** Second\_Procedure1()

**BEGIN**

**INSERT** **INTO** Customer\_Table **DEFAULT** **VALUES;**

SecondSection:**BEGIN**

**DELETE** **FROM** Customer\_Table **WHERE** CUSTOMER\_NUMBER **IS** **NULL;**

**END** SecondSection**;**

**END;**

**CALL** Second\_Procedure1()**;**

**HOW TO DECLARE A VARIABLE:**

**CREATE** **PROCEDURE** Declare\_Procedure2()

**BEGIN**

**DECLARE** var1 INTEGER **DEFAULT** 11111111**;**

**DELETE** **FROM** Customer\_Table **WHERE** CUSTOMER\_NUMBER=:var1**;**

**END;**

**CALL** Declare\_Procedure2()**;**

**HOW TO DECLARE A VARIABLE AND THEN SET THE VARIABLE:**

**CREATE** **PROCEDURE** SetVar\_Procedure2()

**BEGIN**

**DECLARE** var1 INTEGER**;**

**SET** var1=31313131**;**

**DELETE** **FROM** Customer\_Table **WHERE** CUSTOMER\_NUMBER=:var1**;**

**END;**

**CALL** SetVar\_Procedure2()**;**

**AN IN VARIABLE IS PASSED TO THE PROCEDURE DURING THE CALL:**

**CREATE** **PROCEDURE** PassInput\_Procedure2(**IN** var1 INTEGER)

**BEGIN**

**DELETE** **FROM** Customer\_Table **WHERE** CUSTOMER\_NUMBER=:var1**;**

**END;**

**CALL** PassInput\_Procedure2(31323134)**;**

**THE IN, OUT AND INOUT PARAMETERS :**

**CREATE** **PROCEDURE** Test\_Proc2(**IN** var1 BYTEINT,**IN** var2 BYTEINT,**OUT** Msg CHAR(20))

**BEGIN**

**CASE** **WHEN** var1=var2 **THEN** **SET** Msg='They are equal'**;**

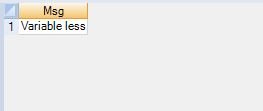
**WHEN** var1<var2 **THEN** **SET** Msg='Variable1 less'**;**

**ELSE** **SET** Msg='Variable1 greater'**;**

**END CASE;**

**END;**

**CALL** Test\_Proc2(1,2,Msg)



**USING IF INSIDE A STORED PROCEDURE:**

**CREATE** **PROCEDURE** Testif\_Proc2(**IN** var1 BYTEINT,**IN** var2 BYTEINT,**OUT** Msg CHAR(20))

**BEGIN**

**IF** var1=var2 **THEN** **SET** Msg='They are equal'**;**

**END IF;**

**IF** var1<var2 **THEN** **SET** Msg='Variable1 less'**;**

**END IF;**

**IF** var1>var2 **THEN** **SET** Msg='Variable1 greater'**;**

**END IF;**

**END;**

**CREATE** **PROCEDURE** Testif1\_Proc2(**IN** var1 BYTEINT,**IN** var2 BYTEINT,**OUT** Msg CHAR(20))

**BEGIN**

**IF** var1=var2 **THEN** **SET** Msg='They are equal'**;**

**END IF;**

**IF** var1<var2 **THEN** **SET** Msg='Variable1 less'**;**

**END IF;**

**IF** var1>var2 **THEN** **SET** Msg='Variable1 greater'**;**

**END IF;**

**ELSE** **SET** Msg='variable 1 is greater'**;**

**END IF;**

**END;**

**USING LOOPS IN STORED PROCEDURE:**

CREATE TABLE My\_Log\_Tbl1(

Cntr Integer

,TheTime Time

)Primary Index(Cntr)**;**

CREATE PROCEDURE Inserter\_Five1( )

LOOPER:BEGIN

DECLARE Cntr INTEGER DEFAULT 0**;**

Loopin:LOOP

SET Cntr=Cntr+1**;**

IF Cntr>5 THEN LEAVE Loopin**;**

END IF**;**

INSERT INTO My\_Log\_Tbl1 VALUES(:Cntr,Time)**;**

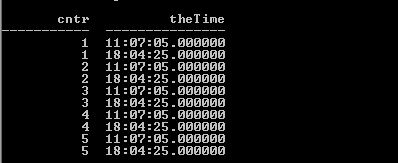
ENDLOOP Loopin**;**

END LOOPER**;**

CALL INSERTER\_FIVE1()

SELECT \* FROM MY\_LOG\_TB11

ORDER BY 1;

****

**NAMING THE FIRST BEGIN AND END:**

CREATE TABLE My\_Log\_Tbl1(

Cntr Integer

,TheTime Time

)Primary Index(Cntr)**;**

CREATE PROCEDURE Inserter\_Five1( )

LOOPER:BEGIN

DECLARE Cntr INTEGER DEFAULT 0**;**

Loopin:LOOP

SET Cntr=Cntr+1**;**

IF Cntr>5 THEN LEAVE Loopin**;**

END IF**;**

INSERT INTO My\_Log\_Tbl1 VALUES(:Cntr,Time)**;**

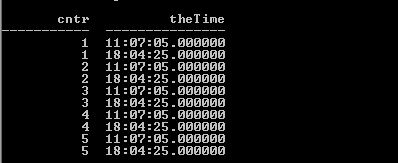
ENDLOOP Loopin**;**

END LOOPER**;**

CALL INSERTER\_FIVE1()

SELECT \* FROM MY\_LOG\_TB11

ORDER BY 1;

****

**USING KEYWORDS LEAVE VS UNTIL LEAVE REPEAT:**

**PROCEDURE ONE:**

CREATE PROCEDURE Ins\_5\_1( )

LOOPER:BEGIN

DECLARE Cntr INTEGER DEFAULT 0**;**

Loopin:LOOP

SET Cntr=Cntr+1**;**

IF Cntr>5 THEN LEAVE Loopin**;**

END IF**;**

INSERT INTO My\_Log\_Tbl1 VALUES(:Cntr,Time)**;**

ENDLOOP Loopin**;**

END LOOPER**;**

**PROCEDURE TWO:**

CREATE PROCEDURE Ins\_5A\_1( )

LOOPER:BEGIN

DECLARE Cntr INTEGER DEFAULT 0**;**

Loopin:REPEAT

SET Cntr=Cntr+1**;**

IF Cntr>5 THEN LEAVE Loopin**;**

END IF**;**

INSERT INTO My\_Log\_Tbl1 VALUES(:Cntr,Time);

UNTIL cntr>4;

ENDREPEAT Loopin**;**

END LOOPER**;**

**STORED PROCEDURE FOR BASIC ASSIGNMENT:**

**CREATE** **MULTISET** **TABLE** TD\_BIM\_FR\_TRNG\_DB.InsProcXYZ1

(col1 INTEGER,

col2 INTEGER)**Primary** **Index**(col1)**;**

**CREATE** **PROCEDURE** TD\_BIM\_FR\_TRNG\_DB.InsProc1()

**BEGIN**

**DECLARE** MyNumber INTEGER **DEFAULT** 0**;**

**SET** MyNumber=1000**;**

MYLOOP:**LOOP**

**SET** MyNumber=MyNumber+1**;**

**IF** MyNumber=2000 **THEN** **LEAVE** MYLOOP**;**

**END IF;**

**INSERT** **INTO** TD\_BIM\_FR\_TRNG\_DB.InsProcXYZ1 **VALUES**(:MyNumber,:MyNumber MOD 250)**;**

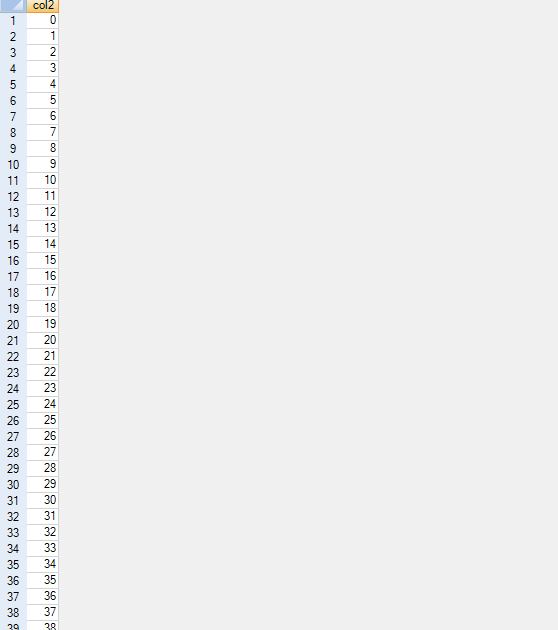
**END LOOP** MYLOOP**;**

**END;**

**CALL** TD\_BIM\_FR\_TRNG\_DB.InsProc1();

**SELECT** **DISTINCT**(col2) **FROM** TD\_BIM\_FR\_TRNG\_DB.InsProcXYZ1

**ORDER** **BY** 1**;**



**STORED PROCEDURE FOR ADVANCED ASSIGNMENT:**

**CREATE** **MULTISET** **TABLE** TD\_BIM\_FR\_TRNG\_DB.InsProc2\_XYZ

(col1 INTEGER,

col2 INTEGER)**Primary** **Index**(col1)**;**

**CREATE** **MULTISET** **TABLE** TD\_BIM\_FR\_TRNG\_DB.InsProc3\_XYZ

(col1 INTEGER,

col2 INTEGER)**Primary** **Index**(col1)**;**

**CREATE** **PROCEDURE** TD\_BIM\_FR\_TRNG\_DB.adv\_assg2()

**BEGIN**

**DECLARE** MyNumber INTEGER **DEFAULT** 0**;**

**SET** MyNumber=1000**;**

MYLOOP:**LOOP**

**SET** MyNumber=MyNumber+1**;**

**IF** MyNumber>2000 **THEN** **LEAVE** MYLOOP**;**

**END IF;**

**INSERT** **INTO** TD\_BIM\_FR\_TRNG\_DB.InsProc2\_XYZ **VALUES**(:MyNumber MOD 500,:MyNumber MOD 100)**;**

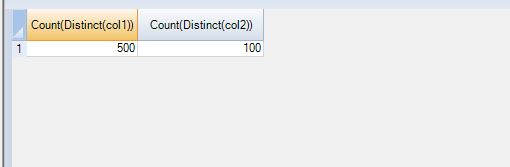
**INSERT** **INTO** TD\_BIM\_FR\_TRNG\_DB.InsProc\_3XYZ **VALUES**(:MyNumber MOD 200,:MyNumber MOD 40)**;**

**END LOOP** MYLOOP**;**

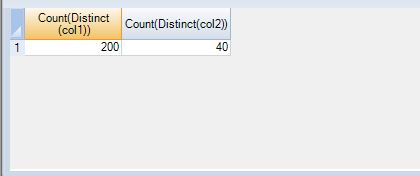
**END;**

**CALL** TD\_BIM\_FR\_TRNG\_DB.adv\_assg2()**;**

**SELECT** COUNT(**DISTINCT**(col1)),COUNT(**DISTINCT**(col1)) **FROM** TD\_BIM\_FR\_TRNG\_DB.InsProc2\_XYZ**;**

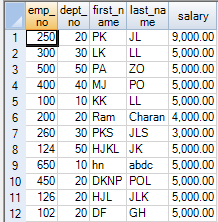
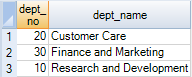


**SELECT** COUNT(**DISTINCT**(col1)),COUNT(**DISTINCT**(col1)) **FROM** TD\_BIM\_FR\_TRNG\_DB.InsProc3\_XYZ**;**

****

**Sub-query Functions**

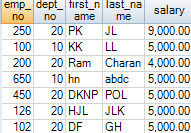
**employee\_table department\_table**

**IN LIST**

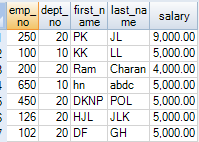
**select** \* **from** employee\_table

**where** dept\_no **IN** (10,20)**;**



**select** \* **from** employee\_table

**where** dept\_no **IN** (10,10,20,20)**;**

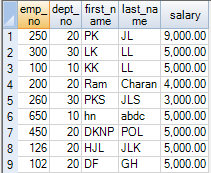


**Working of SUBQUERY**

**select** \* **from** employee\_table

**where** dept\_no **IN** (**select** dept\_no

**from** department\_table )**;**



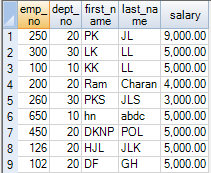
**QUIZ:**

**Using subquery:**

**select** \* **from** employee\_table

**where** dept\_no **IN** (**select** dept\_no

**from** department\_table )**;**



**Using Join:**

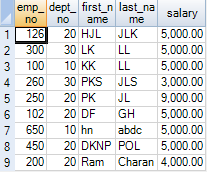
**select** E.\*

**from** employee\_table **as** E

**Inner** **JOIN**

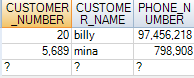
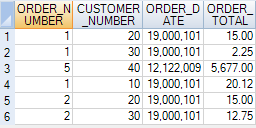
department\_table **as** D

**on** E.dept\_no = D.dept\_no**;**



**Quiz:**

**Cutomer\_table order\_table**

**select** \* **from** customer\_table

**where** customer\_number **IN** (**select** customer\_number

**from** order\_table )**;**



**select** \* **from** customer\_table

**where** customer\_number **IN** (**select** customer\_number

**from** order\_table

**where** order\_total > 10)**;**



**Quiz: Subquery with an aggregate**

**select** \* **from** employee\_table

**where** salary > (**select** avg(salary)

**from** employee\_table)**;**



**Quiz: Correlated Subquery**

**select** \* **from** employee\_table **as** EE

**where** salary > (**select** avg(salary)

**from** employee\_table **as** eeee

**where** ee.dept\_no = eeee.dept\_no)**;**



**Correlated subquery example vs a join with a derived table**

**Correlated subquery**

**select** last\_name, dept\_no,salary

**from** employee\_table **as** EE

**where** salary > (**select** avg(salary)

**from** employee\_table **as** eeee

**where** ee.dept\_no = eeee.dept\_no)**;**



**Join with a derived table**

**select** E.\*, AVGSAL

**from** employee\_table **as** E

**INNER** **JOIN** (**select** dept\_no, avg(salary)

**from** employee\_table

**group** **by** dept\_no)

**as** teratom (depty, avgsal)

**on** dept\_no=depty

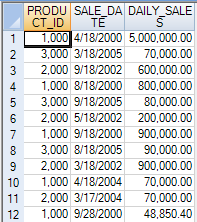
**and** salary> avgsal**;**



**Quiz:**



**Sales\_Table**



**select** \* **from** sales\_table **as** tops

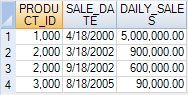
**where** daily\_sales > (

**select** avg ( daily\_sales)

**from** sales\_table **as** bots

**where** tops.product\_id = bots.product\_id)

**order** **by** product\_id, sale\_date**;**



Quiz:



**select** \* **from** sales\_table **as** tops

**where** daily\_sales > (

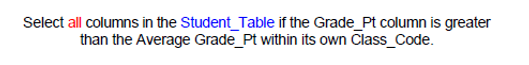
**select** avg ( daily\_sales)

**from** sales\_table **as** bots

**where** tops.sale\_date = bots.sale\_date)

**order** **by** sale\_date**;**

Quiz:



**select** \* **from** student\_table **as** top5

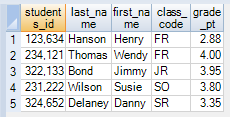
**where** grade\_pt > (

**select** avg ( grade\_pt)

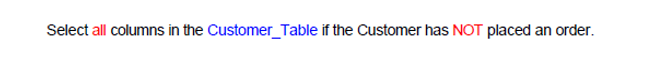
**from** student\_table **as** bot5

**where** top5.class\_code = bot5.class\_code)

**order** **by** class\_code**;**



Quiz:



**select** \* **from** customer\_table

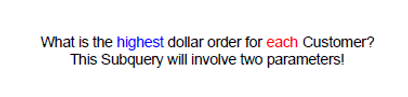
**where** customer\_number **NOT** **IN**

(**select** customer\_number **from** order\_table

**where** customer\_number **is** **not** **null**)**;**



**Subquery with two parameters**



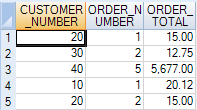
**select** customer\_number, order\_number, order\_total **from** order\_table

**where** (customer\_number,order\_total) **IN**

(**select** customer\_number, max(order\_total)

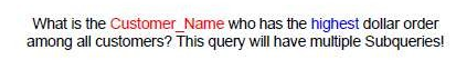
**from** order\_table

**group** **by** 1)**;**



**Triple Subquery**

**Quiz:**



**select** customer\_number **from** customer\_table

**where** customer\_number **IN**

( **select** customer\_number

**from** order\_table

**where** order\_total **IN**

(**select** max(order\_total)

**from** order\_table) )**;**

**Rows return on a Not In with NULL**

**select** customer\_number

**from** customer\_table

**where** customer\_number **NOT** **IN**

(**select** customer\_number

**from** order\_table )**;**



**Using ANY instead of IN**

**select** customer\_number, customer\_name

**from** customer\_table

**where** customer\_number = ANY

(**select** customer\_number

**from** order\_table)**;**



**Using correlated exists**

**select** customer\_number, customer\_name

**from** customer\_table **as** TOP1

**where** EXISTS (**select** \* **from** order\_table **as** BOT1

**where** top1.customer\_number=bot1.customer\_number)**;**

The **correlated not exists**

**select** customer\_number, customer\_name

**from** customer\_table **as** TOP1

**where NOT** EXISTS (**select** \* **from** order\_table **as** BOT1

**where** top1.customer\_number=bot1.customer\_number)**;**

