Subqueries in PROC SQL

- Noncorrelated subqueries

Find job codes for which the group's average salary exceeds the company's average salary. The HAVING clause contains a noncorrelated subquery.

```
proc sql;
   select jobcode,
      avg(salary) as AvgSalary
      format=dollar11.2
   from sasuser.payrollmaster
   group by jobcode
   having avg(salary) >
      (select avg(salary)
         from sasuser.payrollmaster);
quit;
```

- Correlated subqueries

Displays the names of all navigators ("NA") who are also supervisors. The WHERE clause contains a correlated subquery,

```
proc sql;
   select empid, lastname, firstname
      from sasuser.staffmaster
      where 'NA'=
         (select jobcategory
            from sasuser.supervisors
               where staffmaster.empid =
                  supervisors.empid);

quit;
```

Look into the correlated subquery example

EmpID	LastName	FirstName	City	State	PhoneNumber
1919	ADAMS	GERALD	STAMFORD	CT	203/781-1255
1653	ALEXANDER	SUSAN	BRIDGEPORT	CT	203/675-7715
1834	LONG	RUSSELL	NEW YORK	NY	718/384-0040
1350	ARTHUR	BARBARA	NEW YORK	NY	718/383-1549
1401	AVERY	JERRY	PATERSON	NJ	201/732-8787
1499	BAREFOOT	JOSEPH	PRINCETON	NJ	201/812-5665
1101	BAUCOM	WALTER	NEW YORK	NY	212/586-8060
1352	RIVERS	SIMON	NEW YORK	NY	718/383-3345
1402	BLALOCK	RALPH	NEW YORK	NY	718/384-2849
…	…	…	…	…	…

staffmaster

```
proc sql;
   select empid, lastname, firstname
      from sasuser.staffmaster
      where 'NA'=
         (select jobcategory
            from sasuser.supervisors
               where staffmaster.empid =
                  supervisors.empid);
quit;
```

supervisors

EmpID	State	JobCategory
1677	CT	BC
1834	NY	BC
1431	CT	FA
1433	NJ	FA
1983	NY	FA
1385	CT	ME
1420	NJ	ME
1882	NY	ME
1935	CT	NA
1417	NJ	NA
1352	NY	NA
…	…	…

EmpID	LastName	FirstName
1935	FERNANDEZ	KATRINA
1417	NEWKIRK	WILLIAM
1352	RIVERS	SIMON

Number of values returned by a subquery

- A single value
- Multiple values but only for a single column
- In the case of multiple-value noncorrelated subquery, be sure to use one of the following operators:
  - The conditional operator IN
  - A comparison operator that is modified by ANY or ALL
  - The conditional operator EXISTS

Using the IN operator

Find the employees (including empid, name, city and state) who have birthdays in February

```
proc sql;
   select empid, lastname, firstname, city, state
   from sasuser.staffmaster
      where empid IN
         (select empid
            from sasuser.payrollmaster
               where month(dateofbirth)=2);
quit;
```

EmpID	LastName	FirstName	City	State
1403	BOWDEN	EARL	BRIDGEPORT	CT
1404	CARTER	DONALD	NEW YORK	NY
1834	LONG	RUSSELL	NEW YORK	NY
1103	MCDANIEL	RONDA	NEW YORK	NY
1420	ROUSE	JEREMY	PATERSON	NJ
1390	SMART	JONATHAN	NEW YORK	NY

An INNER JOIN can be used to achieve the same result

```
proc sql;
      select staffmaster.empid, lastname, firstname, city, state
          from sasuser.staffmaster, sasuser.payrollmaster
              where staffmaster.empid=payrollmaster.empid
                    and month(dateofbirth)=2;
quit;
```

Note: It is better to use a subquery here since no columns from the payrollmaster table were in the output.

Using comparison operators with ANY and ALL in subqueries

- In a noncorrelated subquery that returns > one value, the WHERE and HAVING clauses in the outer query contains a comparison operator, it must be modified by ANY or ALL.
- If ANY is specified, then the comparison is true if it is true for any one of the values that are returned by the subquery.
- If ALL is specified, then the comparison is true only if it is true for all values that are returned by the subquery.
- The operators ANY and ALL can be used with correlated subqueries, but they are usually used with noncorrelated subqueries.

Using the ANY operator

Using ANY with these common comparison operators: greater than (>), less than (<) and equal to (=)

Comparison Operator with ANY	Outer Query Selects...	Example
> ANY	values that are *greater than any value* returned by the subquery	If the subquery returns the values *20, 30, 40*, then the outer query selects all values that are > *20* (the lowest value that was returned by the subquery).
< ANY	values that are *less than any value* returned by the subquery	If the subquery returns the values *20, 30, 40*, then the outer query selects all values that are < *40* (the highest value that was returned by the subquery).
= ANY	values that are *equal to any value* returned by the subquery	If the subquery returns the values *20, 30, 40*, the outer query selects all values that are = *20* or = *30* or = *40*.

Identify *any* flight attendants at level 1 or level 2 who are older than *any* of the flight attendants at level 3

```
proc sql;
    select empid, jobcode, dateofbirth
        from sasuser.payrollmaster
        where jobcode in ('FA1','FA2')
              and dateofbirth < ANY
                  (select dateofbirth
                       from sasuser.payrollmaster
                       where jobcode='FA3');
quit;
```

EmpID	JobCode	DateOfBirth
1574	FA2	01MAY1958
1475	FA2	19DEC1959
1124	FA1	13JUL1956
1422	FA1	08JUN1962
1368	FA2	15JUN1959
1411	FA2	31MAY1959
1477	FA2	25MAR1962
1970	FA1	29SEP1962
1413	FA2	20SEP1963
...	...	...

Using the MAX or MIN function to replace the ANY operator

Using the ANY operator results in a large number of calculations, which increases processing time. It would be often more efficient to use the MAX or MIN function in the subquery. The previous <ANY comparison operation can be achieved with the following code.

```
proc sql;
   select empid, jobcode, dateofbirth
      from sasuser.payrollmaster
      where jobcode in ('FA1','FA2')
            and dateofbirth < (select MAX(dateofbirth)
                  from sasuser.payrollmaster
                  where jobcode='FA3');
quit;
```

Using the ALL operator

Using ALL with these common comparison operators: greater than (**>**) and less than (**<**).

Comparison Operator with ALL	Sample Values Returned by Subquery	Signifies...
> *ALL*	(20, 30, 40)	> 40 (greater than the highest number in the list)
< ALL	(20, 30, 40)	< 20 (less than the lowest number in the list)

Using the previous query example and substitute ALL for ANY, which identifies level-1 and level-2 flight attendants who are older than all the level-3 attendants.

```
proc sql;
    select empid, jobcode, dateofbirth
        from sasuser.payrollmaster
        where jobcode in ('FA1','FA2')
                and dateofbirth < ALL
                    (select dateofbirth
                        from sasuser.payrollmaster
                        where jobcode='FA3');
quit;
```

EmpID	JobCode	DateOfBirth
1124	FA1	13JUL1956
1415	FA2	12MAR1956

Practice 1

Using the MIN function to replace the ALL operator

It would be more efficient to solve this problem using the MIN function in the subquery instead of the ALL operator.

Hint: you modify the subquery using the MIN function.

Using EXISTS and NOT EXISTS conditional operators

- In the WHERE clause or in the HAVING clause of an outer query, you can use the EXISTS or NOT EXISTS conditional operator to test for the existence or non-existence of a set of values returned by a subquery.
- EXISTS: the condition is true if the subquery returns *at least one row.*
- NOT EXISTS: the condition is true if the subquery returns *no data.*
- The operators EXISTS and NOT EXISTS can be used with both correlated and noncorrelated subqueries but are more often with the correlated subqueries.

Examples: using EXISTS and NOT EXISTS

```
proc sql;
    select lastname, firstname
    from sasuser.flightattendants
    where NOT EXISTS
        (select *
            from sasuser.flightschedule
            where flightattendants.empid=
            flightschedule.empid);
quit;
```

All the flight attendants who not have been scheduled to work

LastName	FirstName
PATTERSON	RENEE
VEGA	FRANKLIN

```
proc sql;
    select lastname, firstname
    from sasuser.flightattendants
    where EXISTS
        (select *
            from sasuser.flightschedule
            where flightattendants.empid=
            flightschedule.empid);
quit;
```

All the flight attendants who have been scheduled to work

LastName	FirstName
ARTHUR	BARBARA
CAHILL	MARSHALL
CARTER	DOROTHY
COOPER	ANTHONY
DEAN	SHARON
DUNLAP	DONNA
EATON	ALICIA
FIELDS	DIANA
…	…

Practice 2

- Business Scenario: Create a report showing `Employee_ID` and `Job_Title` columns of all sales personnel who did not make any sales.
- The table `Sales` contains `Employee_ID` and `Job_Title` columns for all sales personnel. (Download from Blackboard)
- The table `Order_fact` holds information about all sales, and the `Employee_ID` column contains the employee identifier of the staff member who made the sale.
- Build a query with a subquery using NOT EXISTS.

The result of Practice 2

Partial results

Employee_ID	Job_Title
120102	Sales Manager
120103	Sales Manager
120125	Sales Rep. IV
120126	Sales Rep. II
120129	Sales Rep. III
120133	Sales Rep. II
120135	Sales Rep. IV
120137	Sales Rep. III
120139	Sales Rep. II
120140	Sales Rep. I
120142	Sales Rep. III
...	...

Practice 3

- Business Scenario: Create a report listing the employee identifier and first name followed by last name for all managers in Australia.
- In the sasuser library, you have a table, Supervisors, containing EmpID and State for all managers and a table, Employee_Addresses, contains employee names and addresses for all employees.
- Create two exact copies of these two tables in the work library, called EMP_add and Supervisors1, respectively.
- In the Supervisors1 table add a new column named Employee_ID with num data type.
- Copy the value of EmpID (with a char data type) to Employee_ID using the Input() function to convert data type.
- Build the query using a subquery.