

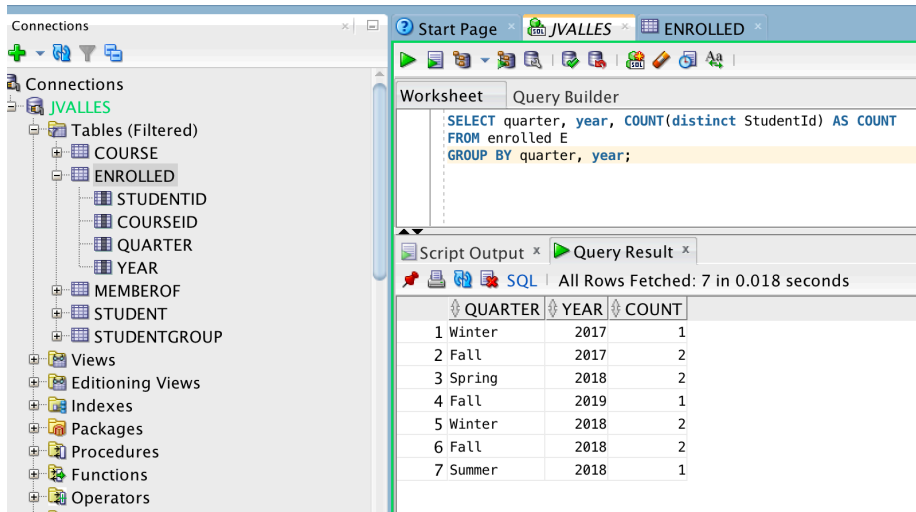
James Valles  
CSC 453  
Assignment – 4

1. Reading done.

2.

a.

```
SELECT quarter, year, COUNT(DISTINCT StudentId) AS COUNT
FROM enrolled E
GROUP BY quarter, year;
```

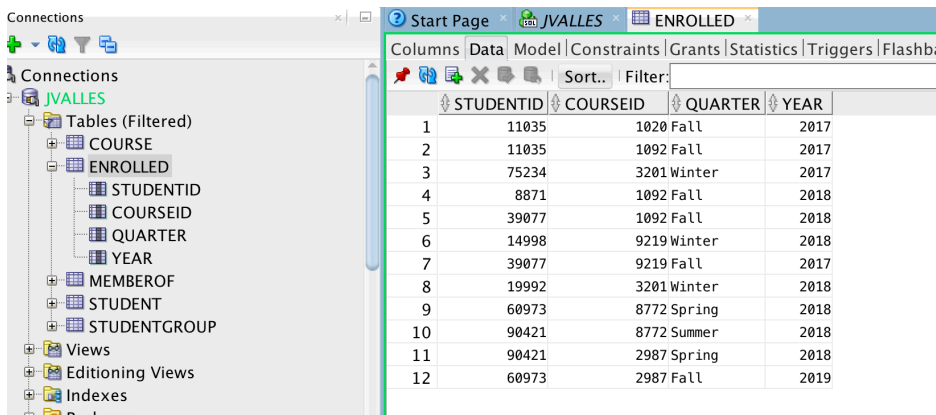


The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'Connections' pane shows a server named 'JVALLES' with a database 'ENROLLED'. The 'Tables (Filtered)' list includes COURSE, ENROLLED, STUDENTID, COURSEID, QUARTER, YEAR, MEMBEROF, STUDENT, and STUDENTGROUP. The 'Query Builder' tab is active, showing the following SQL query:

```
SELECT quarter, year, COUNT(distinct StudentId) AS COUNT
FROM enrolled E
GROUP BY quarter, year;
```

The 'Query Result' tab shows the results of the query, with 7 rows fetched in 0.018 seconds. The results are as follows:

QUARTER	YEAR	COUNT
1 Winter	2017	1
2 Fall	2017	2
3 Spring	2018	2
4 Fall	2019	1
5 Winter	2018	2
6 Fall	2018	2
7 Summer	2018	1



The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'Connections' pane shows a server named 'JVALLES' with a database 'ENROLLED'. The 'Tables (Filtered)' list includes COURSE, ENROLLED, STUDENTID, COURSEID, QUARTER, YEAR, MEMBEROF, STUDENT, and STUDENTGROUP. The 'Query Builder' tab is active, showing the following SQL query:

```
SELECT quarter, year, COUNT(distinct StudentId) AS COUNT
FROM enrolled E
GROUP BY quarter, year;
```

The 'Query Result' tab shows the results of the query, with 12 rows fetched in 0.018 seconds. The results are as follows:

STUDENTID	COURSEID	QUARTER	YEAR
1	11035	1020 Fall	2017
2	11035	1092 Fall	2017
3	75234	3201 Winter	2017
4	8871	1092 Fall	2018
5	39077	1092 Fall	2018
6	14998	9219 Winter	2018
7	39077	9219 Fall	2017
8	19992	3201 Winter	2018
9	60973	8772 Spring	2018
10	90421	8772 Summer	2018
11	90421	2987 Spring	2018
12	60973	2987 Fall	2019

b.

```
SELECT quarter, year, COUNT(distinct StudentId) AS COUNT
FROM enrolled E
GROUP BY quarter, year
ORDER BY year, CASE quarter
    WHEN 'Fall' THEN '0'
    WHEN 'Winter' THEN '1'
    WHEN 'Spring' THEN '2'
    ELSE quarter
end;
```

Connections: JVALLES

Tables (Filtered): COURSE, ENROLLED, STUDENTID, COURSEID, QUARTER, YEAR, MEMBEROF, STUDENT, STUDENTGROUP

Views: Editioning Views, Indexes, Packages, Procedures, Functions, Operators, Queues

Reports

Worksheet: Query Builder

```
SELECT quarter, year, COUNT(distinct StudentId) AS COUNT
FROM enrolled E
GROUP BY quarter, year
ORDER BY year, CASE quarter
WHEN 'Fall' THEN '0'
WHEN 'Winter' THEN '1'
WHEN 'Spring' THEN '2'
ELSE quarter
end;
```

Script Output: Query Result

SQL | All Rows Fetched: 7 in 0.018 seconds

QUARTER	YEAR	COUNT
1 Fall	2017	2
2 Winter	2017	1
3 Fall	2018	2
4 Winter	2018	2
5 Spring	2018	2
6 Summer	2018	1
7 Fall	2019	1

c.

```
SELECT SID, LASTNAME, FIRSTNAME,
(SELECT COUNT(*)
FROM STUDENTGROUP WHERE SID = PRESIDENTID) AS COUNT FROM STUDENT;
```

Connections: JVALLES

Tables (Filtered): COURSE, ENROLLED, STUDENTID, COURSEID, QUARTER, YEAR, MEMBEROF, STUDENT, STUDENTGROUP, GID, NAME, PRESIDENTID, FOUNDED

Views: Editioning Views, Indexes, Packages

Reports: All Reports, Analytic View Reports, Data Dictionary Reports

Start Page: JVALLES, STUDENTGROUP

Worksheet: Query Builder

```
SELECT SID, LASTNAME, FIRSTNAME,
(SELECT COUNT(*)
FROM STUDENTGROUP
WHERE SID = PRESIDENTID) AS COUNT FROM STUDENT;
```

Script Output: Query Result

SQL | All Rows Fetched: 13 in 0.042 seconds

SID	LASTNAME	FIRSTNAME	COUNT
1 90421	Brennigan	Marcus	0
2 14662	Patel	Deepa	0
3 8871	Snowdon	Jonathan	0
4 19992	Starck	Jason	0
5 32105	Johnson	Peter	1
6 11035	Winter	Abigail	0
7 75234	Patel	Prakash	0
8 93321	Snowdon	Jennifer	1
9 14998	Degroff	Jarvis	0
10 57923	Kubik	Dwayne	0
11 58992	Skelly	Trinity	2
12 60973	Krol	Angelo	0
13 39077	Pollard	Joya	0

Columns: Data | Model | Constraints | Grants | Statistics | Triggers | Flashb

Sort: Filter:

GID	NAME	PRESIDENTID	FOUNDED
1	2 Computer Science Society	58992	1999
2	101 Robototics Society	58992	1998
3	221 HerCDM	93321	2003
4	42 DeFrag	32105	2004

3.

a.

The top screenshot shows a database management system interface. On the left, the 'Connections' pane shows a tree view of tables under 'JVALLES'. The 'Tables (Filtered)' list includes COURSE, ENROLLED, STUDENTID, COURSEID, QUARTER, YEAR, MEMBEROF, STUDENT, and STUDENTGROUP. The 'STUDENTGROUP' table is highlighted. On the right, the 'Query Builder' window shows a query:   
`SELECT firstname, lastname  
FROM memberof, student  
WHERE studentid = sid  
GROUP BY firstname, lastname  
HAVING COUNT(SID) >= 2;`  
 Below the query, the 'Script Output' and 'Query Result' tabs are visible. The 'Query Result' tab shows a table with two columns: 'FIRSTNAME' and 'LASTNAME'. It contains two rows: 1 Peter Johnson and 2 Prakash Patel. The bottom screenshot shows the same interface. The 'Connections' pane shows the 'MEMBEROF' table highlighted. The 'Data' tab is selected, showing a table with columns 'STUDENTID', 'GROUPID', and 'JOINED'. It contains 8 rows of data:   

	STUDENTID	GROUPID	JOINED
1	75234	42	2015
2	11035	221	2016
3	93321	221	2015
4	75234	2	2015
5	32105	42	2017
6	32105	2	2018
7	32105	221	2017
8	32105	101	2017

b.

```
select gid from (SELECT gid, nvl(GRDCOUNT, 0) as g, nvl(UGRDCOUNT, 0) as ug FROM
(SELECT sg.GID, s.career, count(s.SID) AS "GRDCOUNT"
FROM memberof m, studentgroup sg, student s
WHERE m.groupID = sg.GID AND s.SID = m.studentId
GROUP BY s.career, sg.GID
HAVING s.career = 'GRD')
natural full outer join
```

```
(SELECT sg.GID, s.career, count(s.SID) AS "UGRDCOUNT"
FROM memberof m, studentgroup sg, student s
WHERE m.groupID = sg.GID AND s.SID = m.studentId
GROUP BY s.career, sg.GID
HAVING s.career= 'UGRD'))
```

```
group by gid
having sum(ug) > sum(g)
;
```

Connections

- JVALLES
  - Tables (Filtered)
    - COURSE
    - ENROLLED
    - MEMBEROF
      - STUDENTID
      - GROUPID
      - JOINED
    - STUDENT
    - STUDENTGROUP
      - GID
      - NAME
      - PRESIDENTID
      - FOUNDED
  - Views
  - Editing Views
  - Indexes
  - Packages
  - Procedures

Reports

- All Reports
- Analytic View Reports

Worksheet Query Builder

```

SELECT gid from (SELECT gid, nvl(GRDCOUNT, 0) as g, nvl(UGRDCOUNT, 0) as ug FROM
(SELECT sg.GID, s.career, count(s.SID) AS "GRDCOUNT"
FROM memberof m, studentgroup sg, student s
WHERE m.groupID = sg.GID AND s.SID = m.studentId
GROUP BY s.career, sg.GID
HAVING s.career = 'GRD')
NATURAL FULL OUTER JOIN
(SELECT sg.GID, s.career, count(s.SID) AS "UGRDCOUNT"
FROM memberof m, studentgroup sg, student s
WHERE m.groupID = sg.GID AND s.SID = m.studentId
GROUP BY s.career, sg.GID
HAVING s.career= 'UGRD'))

GROUP BY gid
HAVING sum(ug) > sum(g);

```

Script Output x Explain Plan x Query Result x

SQL All Rows Fetched: 2 in 0.065 seconds

GID	
1	42
2	2

Start Page x JVALLES x MEMBEROF x

Columns Data Model Constraints Grants Statistics

Sort.. Filter:

	STUDENTID	GROUPID	JOINED
1	75234	42	2015
2	11035	221	2016
3	93321	221	2015
4	75234	2	2015
5	32105	42	2017
6	32105	2	2018
7	32105	221	2017
8	93321	101	2017

Start Page x JVALLES x STUDENT x

Columns Data Model Constraints Grants Statistics Triggers Flashback Dependencies Details Parti

Sort.. Filter:

	LASTNAME	FIRSTNAME	SID	SSN	CAREER	PROGRAM	CITY	STARTED
1	Brennigan	Marcus	90421	987654321	UGRD	INFO-TECH	Evanston	2015
2	Patel	Deepa	14662	(null)	GRD	COMP-SCI	Evanston	2016
3	Snowdon	Jonathan	8871	123123123	GRD	INFO-SYS	Springfield	2017
4	Starck	Jason	19992	789789789	UGRD	INFO-SYS	Springfield	2014
5	Johnson	Peter	32105	123456789	UGRD	COMP-SCI	Chicago	2017
6	Winter	Abigail	11035	111111111	GRD	PHD	Chicago	2016
7	Patel	Prakash	75234	(null)	UGRD	COMP-SCI	Chicago	2014
8	Snowdon	Jennifer	93321	321321321	GRD	COMP-SCI	Springfield	2015
9	Degroff	Jarvis	14998	113311331	GRD	INFO-TECH	Evanston	2016
10	Kubik	Dwayne	57923	979797979	UGRD	COMP-SCI	Springfield	2018
11	Skelly	Trinity	58992	555222555	GRD	PHD	Springfield	2018
12	Krol	Angelo	60973	(null)	UGRD	COMP-SCI	Springfield	2013
13	Pollard	Joya	39077	(null)	GRD	COMP-SCI	Springfield	2017

C.

The screenshot shows a database query tool interface. On the left, a tree view shows the database structure with tables: COURSE, ENROLLED, MEMBEROF, and STUDENT. The ENROLLED table is selected. In the center, the Query Builder shows a query:

```
SELECT STUDENTID, STUDENTID
FROM (SELECT quarter, year, studentid, count(studentid)
FROM enrolled
GROUP BY quarter, year, studentid
HAVING COUNT(studentid) >= 2) NATURAL FULL OUTER JOIN (SELECT courseid, studentid, count(studentid)
FROM enrolled
GROUP BY courseid, studentid
HAVING COUNT(studentid) >= 2);
```

Below the query, the Script Output window shows the results of the query:

STUDENTID	STUDENTID
11035	11035
90421	90421
8871	8871
60973	60973
39077	39077

The screenshot shows the same database query tool interface, but now the ENROLLED table is selected in the tree view. The main window displays the data of the ENROLLED table:

	STUDENTID	COURSEID	QUARTER	YEAR
1	11035	1020	Fall	2017
2	11035	1092	Fall	2017
3	75234	3201	Winter	2017
4	8871	1092	Fall	2017
5	8871	1020	Fall	2017
6	8871	9219	Fall	2017
7	39077	9219	Fall	2017
8	19992	3201	Winter	2018
9	60973	8772	Spring	2018
10	90421	8772	Spring	2018
11	90421	2987	Spring	2018
12	60973	2987	Spring	2018
13	39077	1092	Summer	2018
14	39077	3201	Summer	2018

4.

Can we infer CE  $\rightarrow$  AB? Yes.

C,E

C  $\rightarrow$  D = C,D,E

DE  $\rightarrow$  A, C = C, D, E, A, C

A,C,D  $\rightarrow$  B

Therefore, CE  $\rightarrow$  AB

$\{C, E\}^+ = \{A, B, C, D, E\}$

Can we infer CF  $\rightarrow$  AD? No.

CF  $\rightarrow$  AD

C  $\rightarrow$  D = C,D,F

We can get to D, but not A.

$\{C, F\}^+ = \{C, D, F\}$

### KEYS:

F is not functional dependency (left, right), so must be present in every key.

A,B,C,D, E present in left and right, so will consider all combinations.

To be a key: must be minimal, must determine all attributes in relation.

#### **KEY 1: ACF**

$A, C, F^+ = \{A, B, C, D, E, F\}$

$C \rightarrow D = ACDF$

$ACD \rightarrow B = ABCDF$

F determines itself

$BC \rightarrow AED = ABCDEF$

#### **KEY 2: BCF**

$B, C, F^+ = \{A, B, C, D, E, F\}$

$BC \rightarrow AED = ABCDE$

F determines itself .

Therefore, ABCDEF

#### **KEY 3: CEF**

$C, E, F^+ = \{A, B, C, D, E, F\}$

$C \rightarrow D = CEDF$

$DE \rightarrow AC = ACEDF$

$ACD \rightarrow B = ABCDEF$

#### **KEY 4: DEF**

$D, E, F = \{A, B, C, D, E, F\}$

$DE \rightarrow AC = ACDEF$

$ACD \rightarrow B = ABCDEF$