Module 3.2 – SQL: Clauses & Aggregate Functions

3. SQL: Structured Query Language

- 3.1 Basic SQL Statement
  - Overview
- **3.2 SQL Select Clauses and Aggregate Functions**
  - **Clauses and aggregate functions**
- 3.3 Subqueries
  - WHERE, IN, HAVING, Attribute List (Inline Subquery), Correlated
- 3.4 Functions, Set Operators, Views, DDL, TCL, DCL

DALHOUSIE UNIVERSITY
Inspiring Minds

Module 3.2 – SQL: Clauses and Aggregate Functions

Learning Outcomes

- Understand and explain SQL clauses and aggregate functions
- Create and use SQL statements with clauses and aggregate functions

Textbook Readings

- SQL - Chap 7

Testing*

*Main (but not the only ones) sections of the textbook used for testing are identified in parentheses

- SQL Aggregate Functions (Chap 7.7)

Exercises

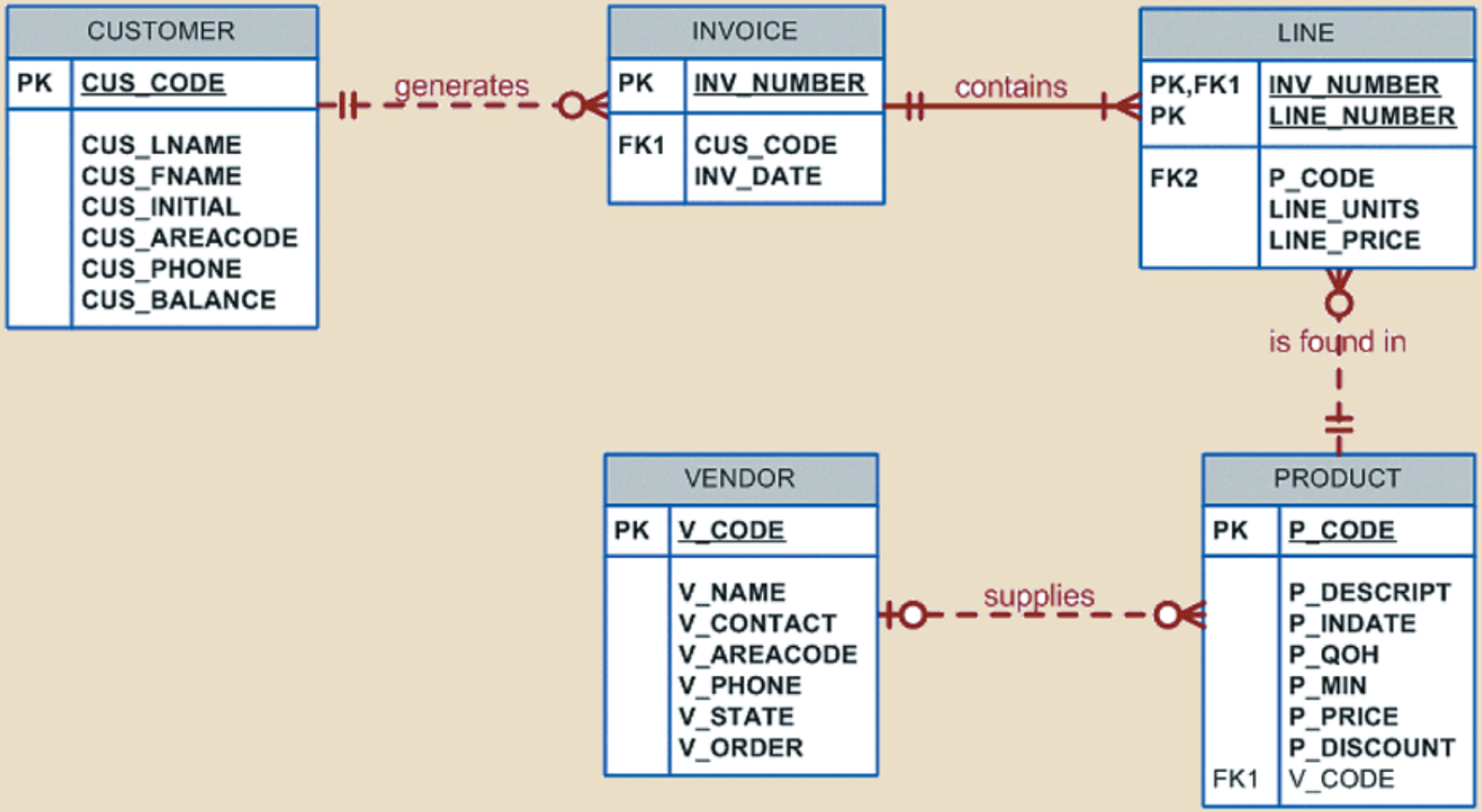
- Exercises

Basic SELECT Statement

- Each clause in a SELECT query performs a specific function
  - SELECT: specifies the attributes to be returned by the query
  - FROM: specifies the table(s) from which the data will be retrieved
  - WHERE: filters the rows of data based on provided criteria
  - ORDER BY: sorts the final query result rows in ascending or descending order based on the values of one or more attributes
  - GROUP BY: groups the rows of data into collections based on sharing the same values in one or more attributes
  - HAVING: filters the groups formed in the GROUP BY clause based on provided criteria

Sample DB used for Intro to SQL

FIGURE 7.1 THE DATABASE MODEL



Aggregate Processing

Aggregate Processing

- We frequently need aggregate information
  - E.g., sum of all salaries for employees for each department
  - List of courses and the total number of students in class
  - List of courses by department and the average marks for each course in the department

- They are derived attributes – that is it is calculated by aggregating over some attribute per each value of another attribute
  - List of departments and the number of course seats per department
    - where course seat == one student registered in one course
  - We want to sum up the count of students registered in each course and then display that sum for each department

Aggregate Processing … continued

- Takes a collection of rows and reduces it to a single row
  - SQL provides useful aggregate functions that count, find minimum and maximum values, calculate averages, etc.
- Aggregate functions
  - Count
  - MIN and MAX
  - SUM and AVG
- Grouping data
  - GROUP BY clause syntax:

```
SELECT          columnlist
FROM            tablelist
[WHERE          conditionlist ]
[GROUP BY   columnlist ]
[ORDER BY   columnlist [ASC | DESC] ];
```

Aggregate Processing … continued

Aggregate Functions

FUNCTION	OUTPUT
COUNT	The number of rows containing non-null values
MIN	The minimum attribute value encountered in a given column
MAX	The maximum attribute value encountered in a given column
SUM	The sum of all values for a given column
AVG	The arithmetic mean (average) for a specified column

Aggregate Processing … continued

- HAVING clause
  - Operates very much like the WHERE clause in the SELECT statement
  - HAVING clause is applied to the output of a GROUP BY operation
  - Syntax:

```
SELECT      columnlist
FROM        tablelist
[WHERE      conditionlist ]
[GROUP BY   columnlist ]
[HAVING     conditionlist ]
[ORDER BY   columnlist [ASC | DESC] ];
```

Aggregate Processing … continued

- SELECT COUNT (p_code)
  FROM product;

CountOfP_CODE
16

- SELECT COUNT(p_price)
  FROM product
  WHERE p_price < 10;

```
SELECT   COUNT ( DISTINCT V_CODE ) AS “Count Distinct”
FROM     PRODUCT;
```

Count Distinct
6

```
SELECT   COUNT (v_code)
FROM     PRODUCT;
```

CountOfV_CODE
14

Contains NULLs in the attribute v_code … NULLS are not counted

Aggregate Processing … continued

- SELECT MAX (p_price) AS MAXPRICE, MIN(p_price) AS MINPRICE
  FROM product;

MAXPRICE	MINPRICE
256.99	4.99

- SELECT SUM (p_qoh * p_price) AS TOTVALUE
  FROM product;

TOTVALUE
15084.52

- SELECT AVG(p_price) AS AVGPRICE
  FROM product;

AVGPRICE
56.42125

FUNCTION	OUTPUT
COUNT	The number of rows containing non-null values
MIN	The minimum attribute value encountered in a given column
MAX	The maximum attribute value encountered in a given column
SUM	The sum of all values for a given column
AVG	The arithmetic mean (average) for a specified column

Grouping Data

- Rows can be grouped into smaller collections using the GROUP BY clause

```
SELECT      columnlist
FROM        tablelist
[ WHERE conditionlist ]
[GROUP BY columnlist ]
[ORDER BY columnlist [ASC | DESC] ];
```

```
SELECT      v_code, AVG (p_price) AS AVGPRICE
FROM        product
GROUP BY    v_ code;
```

{ Average price for products that do not have vendors (v_code is NULL) is also shown. }

V_CODE	AVGPRICE
	10.13
21225	8.47
21231	8.45
21344	12.49
23119	41.97
24288	155.59
25595	89.63

Grouping Data … continued

- SELECT v_code, v_name, COUNT (p_code) AS NUMPROD, AVG(p_price) AS AVGPRICE
  FROM product JOIN vendor ON (product.v_code = vendor.v_code)
  GROUP BY v_code, v_name
  ORDER BY v_name;

V_CODE	V_NAME	NUMPRODS	AVGPRICE
21225	Bryson, Inc.	2	8.47
21231	D&E Supply	1	8.45
21344	Gomez Bros.	3	12.49
23119	Randsets Ltd.	2	41.97
24288	ORDVA, Inc.	3	155.59
25595	Rubicon Systems	3	89.63

- SELECT v_code, v_name, p_qoh, COUNT (p_code), AVG (p_price) AS AVGPRICE
  FROM product JOIN vendor USING (v_code)
  GROUP BY v_code, v_name
  ORDER BY v_name;

{ Error because of p_qoh is not in the GROUP BY clause.
Each attribute in the target list is either included in an aggregate (grouped) function or appears in the GROUP BY clause. }

Grouping Data … continued

- SELECT v_code, v_name, SUM (p_qoh) AS TOTALQTY,
  COUNT (p_code) AS NUMPROD, AVG(p_price) AS AVGPRICE
  FROM product JOIN vendor USING (v_code)
  GROUP BY v_code, v_name
  ORDER BY v_name;

V_CODE	V_NAME	TOTALQTY	NUMPRODS	AVGPRICE
21225	Bryson, Inc.	195	2	8.47
21231	D&E Supply	237	1	8.45
21344	Gomez Bros.	93	3	12.49
23119	Randsets Ltd.	38	2	41.97
24288	ORDVA, Inc.	25	3	155.59
25595	Rubicon Systems	38	3	89.63

- SELECT v_code, v_name, p_qoh, COUNT (p_code) AS NUMPROD, AVG(p_price) AS AVGPRICE
  FROM product JOIN vendor USING (v_code)
  GROUP BY v_code, v_name, p_qoh
  ORDER BY v_name;

V_CODE	V_NAME	P_QOH	NUMPRODS	AVGPRICE
21225	Bryson, Inc.	23	1	9.95
21225	Bryson, Inc.	172	1	6.99
21231	D&E Supply	237	1	8.45
21344	Gomez Bros.	18	1	17.49
21344	Gomez Bros.	32	1	14.99
21344	Gomez Bros.	43	1	4.99
23119	Randsets Ltd.	15	1	39.95
23119	Randsets Ltd.	23	1	43.99
24288	ORDVA, Inc.	6	1	99.87

Having Clause

- HAVING clause
  - Operates very much like the WHERE clause in the SELECT statement
  - HAVING clause is applied to the output of a GROUP BY operation
  - Syntax:

```
SELECT      columnlist
FROM        tablelist
[WHERE      conditionlist ]
[GROUP BY   columnlist ]
[HAVING     conditionlist ]
[ORDER BY   columnlist [ASC | DESC] ];
```

Having Clause … continued

- SELECT V_CODE, COUNT(P_CODE) AS NUMPRODS
  FROM RODUCT
  GROUP BY V_CODE
  HAVING AVG (P_PRICE) < 10
  ORDER BY V_CODE;

V_CODE	NUMPRODS
21225	2
21231	1

- SELECT V_CODE, V_NAME, SUM(P_QOH * P_PRICE) AS TOTCOST
  FROM PRODUCT JOIN VENDOR ON (PRODUCT.V_CODE = VENDOR.V_CODE)
  WHERE P_DISCOUNT > 0
  GROUP BY V_CODE, V_NAME
  HAVING (SUM(P_QOH * P_PRICE) > 500)
  ORDER BY SUM(P_QOH * P_PRICE) DESC;

- Note the syntax used in the HAVING and ORDER BY clauses;
  - in both cases, you should specify the column expression (formula) used in the SELECT statement's column list, rather than the column alias (TOTCOST). Some RDBMSs allow you to replace the column expression with the column alias, while others do not.

SUPPLIER-PARTS DB USED IN EXAMPLES

SUPPLIER-PARTS DB

S

sNo (PK)	sName	sStatus	sCity
…	…	…	…

P

pNo (PK)	pName	pColor	pWeight
…	…	…	…

SP

sNo (PK) (FK)	pNo (PK) (FK)	qty
…	…	…

Notation: PK … Primary Key
FK … Foreign Key

Each tuple (SNO, PNO, QTY) in SP is considered to be an order for QTY units of part PNO from the supplier SNO.

Examples … continued

- Display information about parts, such that the result is ordered by color (ascending) in and then by weight (descending).

```
SELECT      p.*
FROM        p
ORDER BY    pColor ASC, pWeight DESC;
```

pNo	pName	pColor	pWeight
P2	Bolt	Blue	17
P5	Cam	Blue	12
P3	Screw	Green	17
P4	Screw	Red	14
P1	Nut	Red	12

Examples … continued

- Find the number of parts for each color. Display the information in reverse order by color.

```
SELECT      pColor, COUNT(pColor)
FROM        p
GROUP BY    pColor
ORDER BY    pColor DESC;
```

pColor	COUNT(pColor)
Red	2
Green	1
Blue	2

Examples … continued

- For each supplier, find the count, sum, minimum, maximum, and average qty on order. Display information in the ascending order by the sum of quantities.
- SELECT sNo, COUNT(sNo), SUM(qty), MIN(qty), AVG(qty), MAX(qty)
- FROM sp
- GROUP BY sNo
- ORDER BY SUM(qty);

sNo	COUNT(sNo)	SUM(qty)	MIN(qty)	AVG(qty)	MAX(qty)
S5	1	200	200	200.0000	200
S3	3	500	100	166.6667	200
S4	2	600	200	300.0000	400
S2	2	700	300	350.0000	400
S1	5	1200	100	240.0000	400

Examples … continued

- For each London supplier, find the count, sum, minimum, maximum, and average qty on order. Display information in the ascending order by the sum of quantities.

```
SELECT      sNo, COUNT(sNo), SUM(qty), MIN(qty), AVG(qty), MAX(qty)
FROM        sp JOIN s USING (sNo)
WHERE       sCity = 'London'
GROUP BY    sNo
ORDER BY    SUM(qty);
```

sNo	COUNT(sNo)	SUM(qty)	MIN(qty)	AVG(qty)	MAX(qty)
S4	2	600	200	300.0000	400
S2	2	700	300	350.0000	400
S1	5	1200	100	240.0000	400

Examples … continued

- For each London supplier find the total number of orders, total quantity on order, minimum, maximum, and average qty on order for red parts.

```
SELECT      sNo, COUNT(sNo), SUM(qty), MIN(qty), AVG(qty), MAX(qty)
FROM        sp JOIN p USING (pNo) JOIN s USING (sNo)
WHERE       sCity = 'London' AND pColor = 'Red'
GROUP BY    sNo
ORDER BY    SUM(qty);
```

sNo	COUNT(sNo)	SUM(qty)	MIN(qty)	AVG(qty)	MAX(qty)
S2	1	300	300	300.0000	300
S1	2	500	200	250.0000	300

Examples … continued

- For each London supplier find the total number of orders, total quantity on order, minimum, maximum, and average qty on order for red parts. However, display information only for those suppliers for which the average qty on order was greater than 270.
- SELECT sNo, COUNT(sNo), SUM(qty), MIN(qty), AVG(qty), MAX(qty)
- FROM sp JOIN p USING (pNo) JOIN s USING (sNo)
- WHERE sCity = 'London' AND pColor = 'Red'
- GROUP BY sNo
- HAVING AVG(qty) > 270
- ORDER BY SUM(qty);

sNo	COUNT(sNo)	SUM(qty)	MIN(qty)	AVG(qty)	MAX(qty)
S2	1	300	300	300.0000	300

Questions and Answers (Q/A)