Data 604: Data Management

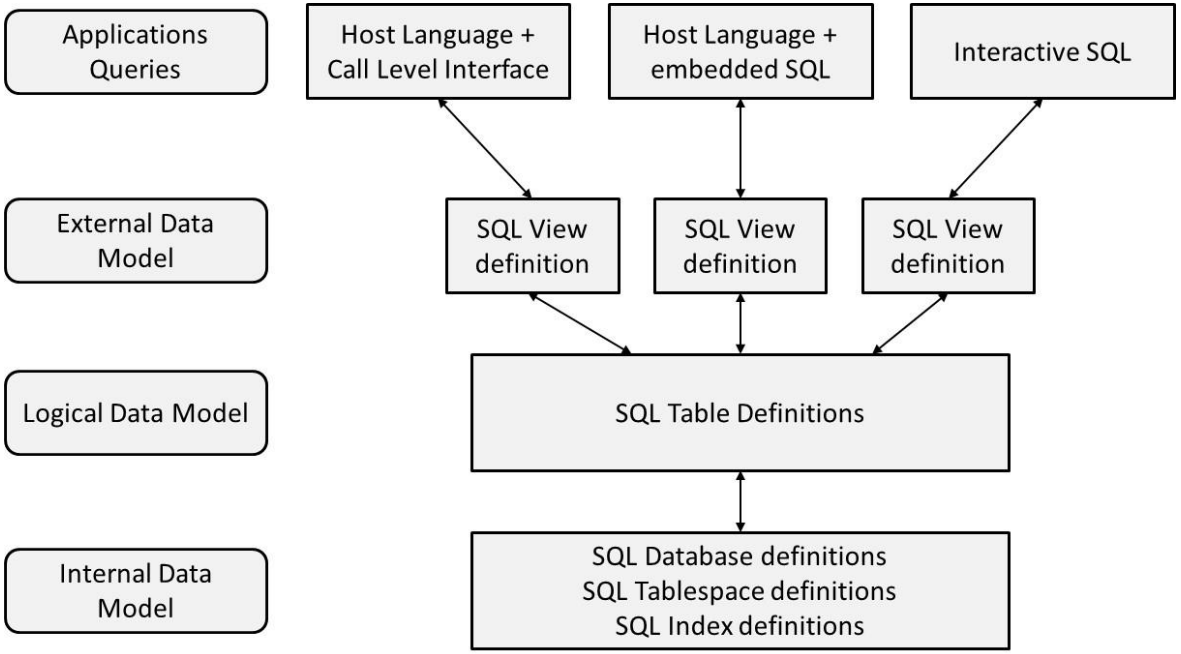
Lecture 4 – Topics:

- Basic SQL
  - Introduction to SQL
  - History of SQL
  - SQL Environment
  - Basic SQL Queries

Learning Objectives

- Write single- and multiple-table queries using SQL commands
- Define three types of join commands and use SQL to write these commands
- Write noncorrelated and correlated subqueries and know when to write each
- Write queries to create views
- Understand common uses of database triggers and stored procedures
- Understand Boolean and Control and Flow operators
- Understand difference between Loop and Batch operations and when to use each

Three-Level Database Architecture



SQL

- Structured Query Language – often pronounced "Sequel"
- The standard for Relational Database Management Systems (RDBMS)
- RDBMS: A database management system that manages data as a collection of tables in which all relationships are represented by common values in related tables

Key Characteristics of SQL

- First version, SQL-86 in 1986, most recent version in 2016 (SQL:2019)
- Accepted by the American National Standards Institute (ANSI) in 1986 and by the International Organization for Standardization (ISO) in 1987
- Each vendor provides own implementation (SQL dialect) of SQL
- Set-oriented and declarative
- Free form language
- Case insensitive
- Can be used interactively from a command prompt or executed by a program

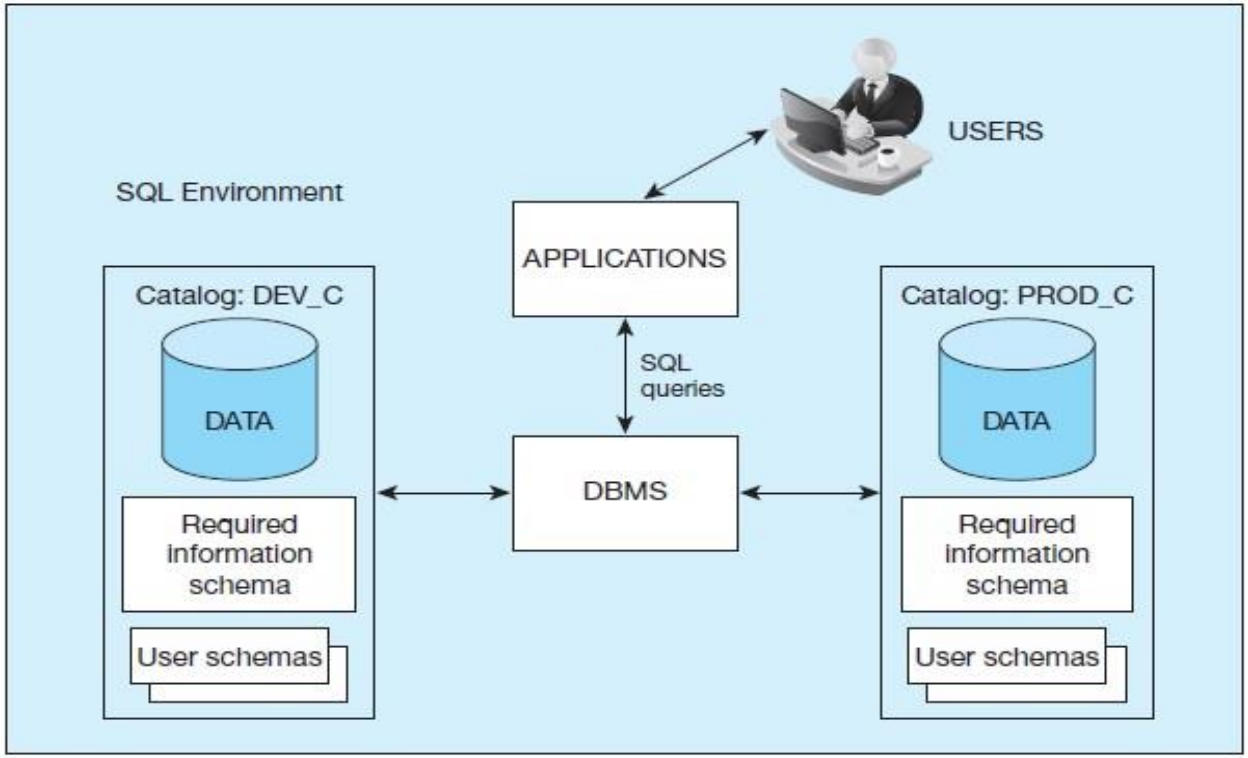
Original Purpose of SQL Standard

- Specify syntax/semantics for data definition and manipulation
- Define data structures and basic operations
- Enable portability of database definition and application modules
- Specify minimal (level 1) and complete (level 2) standards
- Allow for later growth/enhancement to standard (referential integrity, transaction management, user-defined functions, extended join operations, national character sets)

SQL Environment

- Catalog
  - A set of schemas that constitute the description of a database
- Schema
  - The structure that contains descriptions of objects created by a user (base tables, views, constraints)
- Data Definition Language (DDL)
  - Commands that define a database, including creating, altering, and dropping tables and establishing constraints
- Data Manipulation Language (DML)
  - Commands that maintain and query a database
- Data Control Language (DCL)
  - Commands that control a database, including administering privileges and committing data

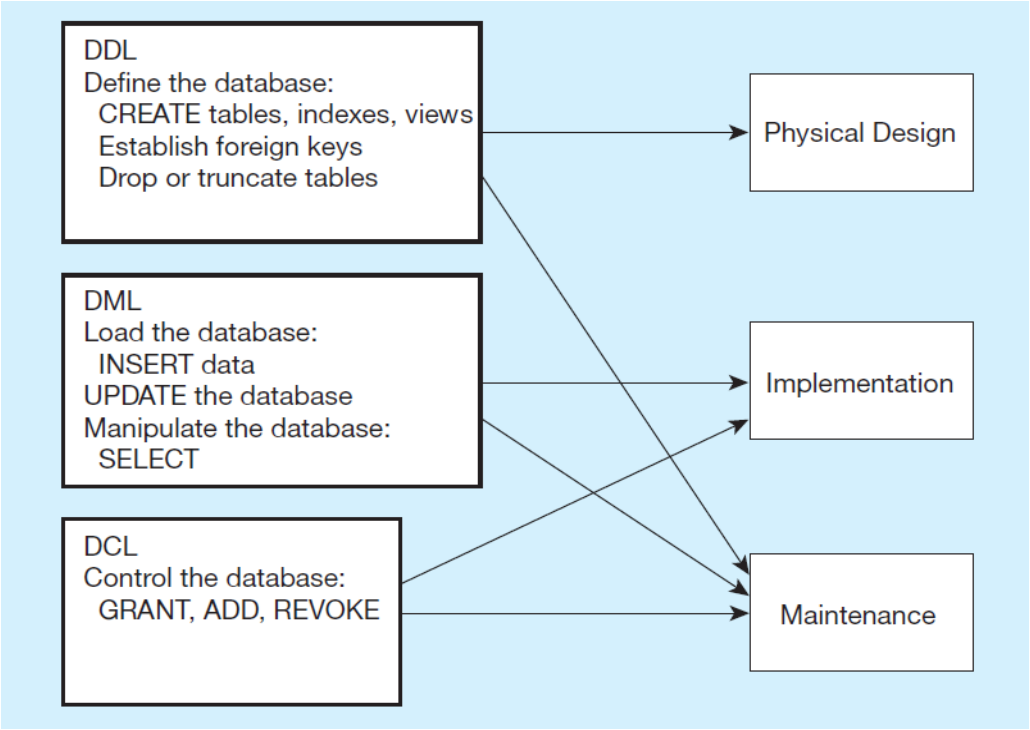
SQL Environment, as Described by SQL:2016 Standard



SQL Datatypes

- Strings
  - CHARACTER (n), VARYING CHARACTER (n)
- Binary
  - Binary Large Object (BLOB)
- Number
  - Numeric (precision, scale), Decimal (p, s), Integer
- Temporal
  - Timestamp, Timestamp with local time zone
- Boolean
  - True or False values

DDL, DML, DCL, and Database Development Process



SQL Data Definition Language

- Key DDL concepts
- DDL Example
- Referential Integrity Constraints
- DROP and ALTER command

Key DDL Concepts

- SQL Schema: grouping of tables and other database objects such as views, constraints and indexes which logically belong together

```
CREATE SCHEMA PURCHASE AUTHORIZATION
BBAESENS
```

- SQL table implements a relation from the relational model

```
CREATE TABLE PRODUCT …
CREATE TABLE PURCHASE.PRODUCT …
```

Key DDL Concepts

Data Type	Description
CHAR(n)	Holds a fixed length string with size n
VARCHAR(n)	Holds a variable length string with maximum size n
SMALLINT	Small integer (no decimal) between -32768 to 32767
INT	Integer (no decimal) between -2147483648 to 2147483647
FLOAT(n,d)	Small number with a floating decimal point. The total maximum number of digits is n with a maximum of d digits to the right of the decimal point.
DOUBLE(n,d)	Large number with a floating decimal point. The total maximum number of digits is n with a maximum of d digits to the right of the decimal point.
DATE	Date in format YYYY-MM-DD
DATETIME	Date and time in format YYYY-MM-DD HH:MI:SS
TIME	Time in format HH:MI:SS
BOOLEAN	True or False
BLOB	Binary Large Object (e.g. image, audio, video)

Steps in Table Creation

1. Identify data types for attributes
2. Identify columns that can and cannot be null
3. Identify columns that must be unique (candidate keys)
4. Identify primary key–foreign key mates
5. Determine default values
6. Identify constraints on columns (domain specifications)
7. Create the table and associated indexes

CREATE TABLE Syntax

```
CREATE TABLE tablename
( {column definition      [table constraint] } . , . .
[ON COMMIT {DELETE | PRESERVE} ROWS] );

where column definition ::=
column_name
        {domain name | datatype [(size)] }
        [column_constraint_clause. . .]
        [default value]
        [collate clause]

and table constraint ::=
        [CONSTRAINT constraint_name]
        Constraint_type [constraint_attributes]
```

Defining Attributes and Data Types

```
CREATE TABLE Product_T
        (ProductID                  NUMBER(11,0)      NOT NULL,
         ProductDescription         VARCHAR2(50),
         ProductFinish              VARCHAR2(20)
                CHECK (ProductFinish IN ('Cherry', 'Natural Ash', 'White Ash',
                                         'Red Oak', 'Natural Oak', 'Walnut')),
         ProductStandardPrice       DECIMAL(6,2),
         ProductLineID              INTEGER,
CONSTRAINT Product_PK PRIMARY KEY (ProductID));
```

Key DDL Concepts

- Column constraints
  - **PRIMARY KEY** constraint defines the primary key of the table
  - **FOREIGN KEY** constraint defines a foreign key of a table
  - **UNIQUE** constraint defines an alternative key of a table
  - **NOT NULL** constraint prohibits NULL values for a column
  - **DEFAULT** constraint sets a default value for a column
  - **CHECK** constraint defines a constraint on the column values

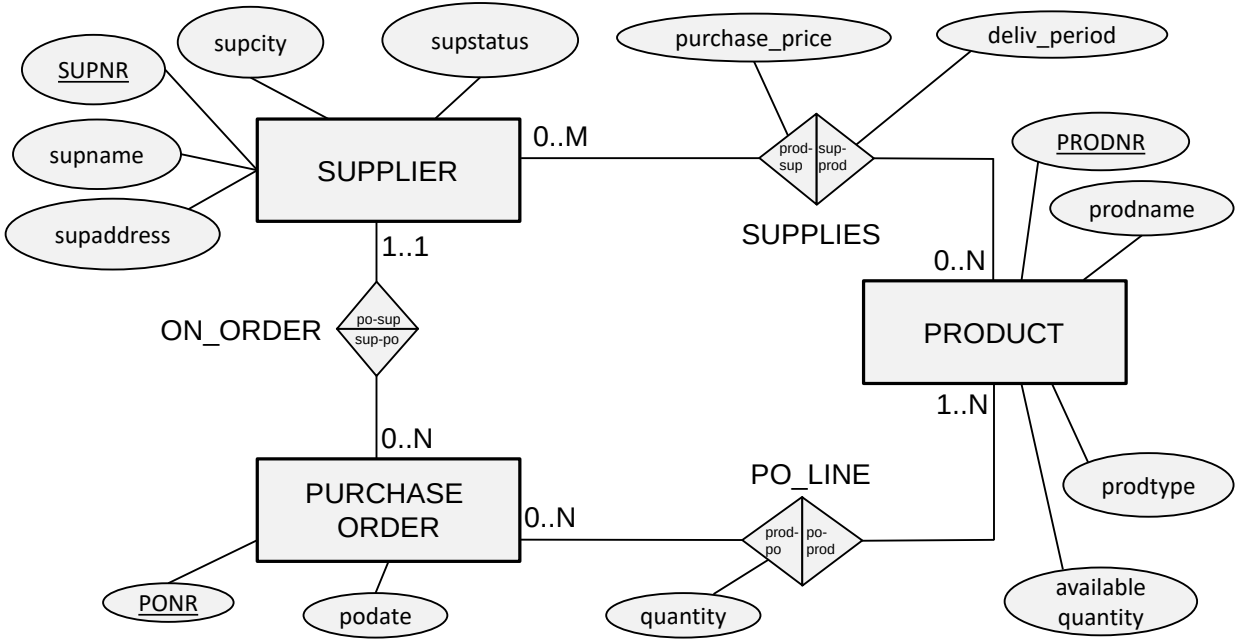
Primary and Foreign Keys

```
CREATE TABLE Customer_T
                    (CustomerID                 NUMBER(11,0)        NOT NULL,
                     CustomerName               VARCHAR2(25)        NOT NULL,
                     CustomerAddress            VARCHAR2(30),
                     CustomerCity               VARCHAR2(20),
                     CustomerState              CHAR(2),
                     CustomerPostalCode         VARCHAR2(9),
CONSTRAINT Customer_PK PRIMARY KEY (CustomerID));
```

Primary key of parent table

```
CREATE TABLE Order_T
                    (OrderID                    NUMBER(11,0)        NOT NULL,
                     OrderDate                  DATE DEFAULT SYSDATE,
                     CustomerID                 NUMBER(11,0),
CONSTRAINT Order_PK PRIMARY KEY (OrderID),
CONSTRAINT Order_FK FOREIGN KEY (CustomerID) REFERENCES Customer_T(CustomerID));
```

DDL Example



DDL Example

SUPPLIER(SUPNR, SUPNAME, SUPADDRESS, SUPCITY, SUPSTATUS)

PRODUCT(PRODNR, PRODNAME, PRODTYPE, AVAILABLE_QUANTITY)

SUPPLIES(*SUPNR*, *PRODNR*, PURCHASE_PRICE, DELIV_PERIOD)

PURCHASE_ORDER(PONR, PODATE, *SUPNR*)

PO_LINE(*PONR*, *PRODNR*, QUANTITY)

DDL Example

```
CREATE TABLE SUPPLIER
(SUPNR CHAR(4) NOT NULL PRIMARY KEY,
 SUPNAME VARCHAR(40) NOT NULL,
 SUPADDRESS VARCHAR(50),
 SUPCITY VARCHAR(20),
 SUPSTATUS SMALLINT)

CREATE TABLE PRODUCT
(PRODNR CHAR(6) NOT NULL PRIMARY KEY,
 PRODNAME VARCHAR(60) NOT NULL,
 CONSTRAINT UC1 UNIQUE(PRODNAME),
 PRODTYPE VARCHAR(10),
 CONSTRAINT CC1 CHECK(PRODTYPE IN ('white', 'red', 'rose','sparkling')),
 AVAILABLE_QUANTITY INTEGER)
```

DDL Example

```
CREATE TABLE SUPPLIES
(SUPNR CHAR(4) NOT NULL,
 PRODNR CHAR(6) NOT NULL,
 PURCHASE_PRICE DECIMAL(8,2),
 DELIV_PERIOD TIME,
 PRIMARY KEY (SUPNR, PRODNR),
 FOREIGN KEY (SUPNR) REFERENCES SUPPLIER (SUPNR)
 ON DELETE CASCADE ON UPDATE CASCADE,
 FOREIGN KEY (PRODNR) REFERENCES PRODUCT (PRODNR)
 ON DELETE CASCADE ON UPDATE CASCADE)
```

DDL Example

```
CREATE TABLE PURCHASE_ORDER
(PONR CHAR(7) NOT NULL PRIMARY KEY,
 PODATE DATE,
 SUPNR CHAR(4) NOT NULL,
 FOREIGN KEY (SUPNR) REFERENCES SUPPLIER (SUPNR)
 ON DELETE CASCADE ON UPDATE CASCADE)

CREATE TABLE PO_LINE
 (PONR CHAR(7) NOT NULL,
  PRODNR CHAR(6) NOT NULL,
  QUANTITY INTEGER,
  PRIMARY KEY (PONR, PRODNR),
  FOREIGN KEY (PONR) REFERENCES PURCHASE_ORDER (PONR)
  ON DELETE CASCADE ON UPDATE CASCADE,
  FOREIGN KEY (PRODNR) REFERENCES PRODUCT (PRODNR)
  ON DELETE CASCADE ON UPDATE CASCADE)
```

Data Integrity Constraint

- Referential integrity – constraint that ensures that foreign key values of a table must match primary key values of a related table in relationships
  - Restricting:
    - Deletes of primary records
    - Updates of primary records
    - Inserts of dependent records

Referential Integrity Constraints

- Foreign key has the same domain as the primary key it refers to and either occurs as a value of it or NULL
- What happens to foreign keys when primary key is updated or deleted?
- Options:
  - **ON UPDATE/DELETE CASCADE**: update/removal should be cascaded to all referring tuples
  - **ON UPDATE/DELETE RESTRICT:** update/removal is halted if referring tuples exist
  - **ON UPDATE/DELETE SET NULL:** foreign keys in the referring tuples are set to NULL
  - **ON UPDATE/DELETE SET DEFAULT:** foreign keys in the referring tuples are set to their default value

Referential Integrity Constraints

Supplier

SUPNR	SUPNAME	SUPADDRESS	SUPCITY	SUPSTATUS
21	Deliwines	240, Avenue of the Americas	New York	20
32	Best Wines	660, Market Street	San Francisco	90
37	**Ad Fundum**	**82, Wacker Drive**	**Chicago**	**95**
52	Spirits & co.	928, Strip	Las Vegas	NULL
68	The Wine Depot	132, Montgomery Street	San Francisco	10
69	Vinos del Mundo	4, Collins Avenue	Miami	92

Supplies

SUPNR	*PRODNR*	PURCHASE_PRICE	DELIV_PERIOD
37	0178	16.99	4
37	0185	32.99	3
37	0468	14.00	1
37	0795	20.99	3

Purchase_Order

PONR	PODATE	*SUPNR*
1511	2015-03-24	37
1513	2015-04-11	37
1523	2015-04-19	37
1577	2015-05-10	37
1594	2015-05-13	37

ALTER Command

- **ALTER** statement can be used to modify table column definitions
- Examples:

```
ALTER TABLE PRODUCT ADD PRODIMAGE BLOB
    in SQL Server use VARBINARY(MAX)

ALTER TABLE SUPPLIER ALTER SUPSTATUS SET DEFAULT '10'
```

DROP Command

- **DROP** command can be used to drop or remove database objects
  - can be combined with CASCADE and RESTRICT
- Examples:

```
DROP SCHEMA PURCHASE CASCADE
DROP SCHEMA PURCHASE RESTRICT
DROP TABLE PRODUCT CASCADE
DROP TABLE PRODUCT RESTRICT
```

SQL Data Manipulation Language (SQL DML)

- SQL INSERT Statement
- SQL DELETE Statement
- SQL UPDATE Statement
- SQL SELECT Statement

Insert Statement

- Adds one or more rows to a table
- Inserting into a table:
- Inserting a record that has some null attributes requires identifying the fields that actually get data:
- Inserting from another table:

Identity Column

Inserting into a table does not require explicit customer ID entry or field list.

```
INSERT INTO CUSTOMER_T VALUES ( 'Contemporary Casuals', '1355 S. Himes Blvd.',
'Gainesville', 'FL', 32601);
```

```
CREATE TABLE Customer_T
(CustomerID INTEGER GENERATED ALWAYS AS IDENTITY
     (START WITH 1
     INCREMENT BY 1
     MINVALUE 1
     MAXVALUE 10000
     NO CYCLE),
CustomerName            VARCHAR2(25) NOT NULL,
CustomerAddress         VARCHAR2(30),
CustomerCity            VARCHAR2(20),
CustomerState           CHAR(2),
CustomerPostalCode      VARCHAR2(9),
CONSTRAINT Customer_PK PRIMARY KEY (CustomerID));
```

Delete Statement

- Removes rows from a table
- Delete certain rows

```
DELETE FROM Product
WHERE ProductID = 24
```

- Delete all rows
  - DELETE FROM Product
  - Truncate Table Product (does not log deletions)

UPDATE Statement

- Modifies data in existing rows
  - UPDATE Product_T SET ProductStandardPrice = 775 WHERE ProductID = 7;

SELECT Statement

- Used for queries on single or multiple tables
- Clauses of the SELECT statement:
  - SELECT: List the columns (and expressions) to be returned from the query
  - FROM: Indicate the table(s) or view(s) from which data will be obtained
  - WHERE: Indicate the conditions under which a row will be included in the result
  - GROUP BY: Indicate categorization of results
  - HAVING: Indicate the conditions under which a category (group) will be included
  - ORDER BY: Sorts the result according to specified criteria

SELECT Example

- Find products with standard price less than $275

```
SELECT ProductDescription, ProductStandardPrice
  FROM Product_T
      WHERE ProductStandardPrice < 275;
```

- Comparison operators include
  - = Equal to
  - > Greater than
  - >= Greater than or equal to
  - < Less than
  - <= Less than or equal to
  - <> Not equal to
  - != Not equal to

Alias

- Alias is an alternative column or table name

```
SELECT CUST.CUSTOMERNAME AS NAME,
       CUST.CUSTOMERADDRESS
FROM CUSTOMER_V CUST
            WHERE NAME = 'Home Furnishings';
```

Function

- Using the COUNT aggregate function to find totals

```
SELECT COUNT(*) FROM ORDERLINE_T
WHERE ORDERID = 1004;
```

- Note: With aggregate functions you can't have single-valued columns included in the SELECT clause, unless they are included in the GROUP BY clause.

Boolean Operators

- AND, OR, and NOT Operators for customizing conditions in WHERE clause

```
SELECT ProductDescription, ProductFinish, ProductStandardPrice
FROM Product_T
WHERE ProductDescription LIKE '%Desk'
OR ProductDescription LIKE '%Table'
AND ProductStandardPrice > 300;
```

Boolean Operators with Parenthesis

With parentheses…these override the normal precedence of Boolean operators

```
SELECT ProductDescription, ProductFinish,
ProductStandardPrice
FROM Product_T;
WHERE (ProductDescription LIKE '%Desk'
OR ProductDescription LIKE '%Table')
AND ProductStandardPrice > 300;
```

With parentheses, you can override normal precedence rules. In this case parentheses make the OR take place before the AND.

ORDER BY Clause

- Sort the results first by STATE, and within a state by the CUSTOMER NAME

```
SELECT CustomerName, CustomerCity, CustomerState
FROM Customer_T
WHERE CustomerState IN ('FL', 'TX', 'CA', 'HI')
ORDER BY CustomerState, CustomerName;
```

- Note: The IN operator in this example allows you to include rows whose CustomerState value is either FL, TX, CA, or HI. It is more efficient than separate OR conditions.

GROUP BY

- For use with aggregate functions
  - **Scalar aggregate**: single value returned from SQL query with aggregate function
  - **Vector aggregate**: multiple values returned from SQL query with aggregate function (via GROUP BY)

```
SELECT CustomerState, COUNT (CustomerState)
FROM Customer_T
GROUP BY CustomerState;
```

- You can use single-value fields with aggregate functions if they are included in the GROUP BY clause

HAVING

- For use with GROUP BY

```
SELECT CustomerState, COUNT (CustomerState)
FROM Customer_T
GROUP BY CustomerState
HAVING COUNT (CustomerState) > 1;
```

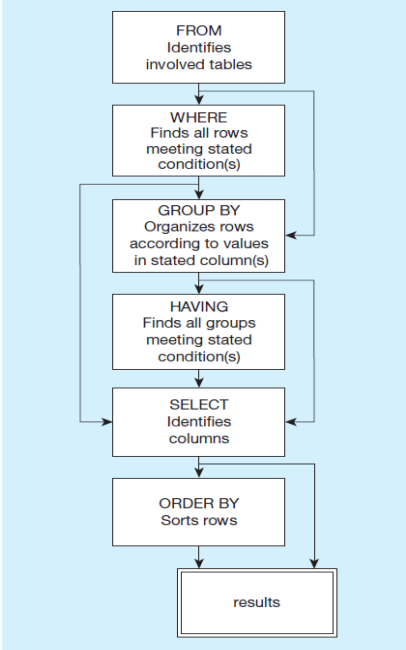
- Like a WHERE clause, but it operates on groups (categories), not on individual rows. Here, only those groups with total numbers greater than 1 will be included in the final result.

Query with WHERE and Having

```
SELECT ProductFinish, AVG (ProductStandardPrice)
FROM Product_T
WHERE ProductFinish IN ('Cherry', 'Natural Ash', 'Natural Maple',
'White Ash')
GROUP BY ProductFinish
HAVING AVG (ProductStandardPrice) < 750
ORDER BY ProductFinish;
```

SQL Statement Processing Order

```
SELECT [ALL/DISTINCT] column_list
FROM table_list
[WHERE conditional expression]
[GROUP BY group_by_column_list]
[HAVING conditional expression]
[ORDER BY order_by_column_list]
```



- (based on van der Lans, 2006, p. 100)

SQL SELECT Statement – In Depth view

- Overview
- Simple Queries
- Queries with Aggregate Functions
- Queries with GROUP BY/HAVING
- Queries with ORDER BY
- Queries with IN
- Join Queries
- Nested Queries
- Correlated Queries
- Queries with EXISTS
- Queries with subqueries in FROM/WHERE
- Queries with Set operations

Overview

```
SELECT component
FROM component
[WHERE component]
[GROUP BY component]
[HAVING component]
[ORDER BY component]
```

Predicates (TSQL p.49)

- Three valued logic expression (true, false, null)
- Examples WHERE, HAVING, IN, BETWEEN, LIKE
- EXISTS is a predicate that differs from the others as it is a two valued logic expression (true, false) (p.145)

Operators for Where Clause

- Comparison operators are
  - = Equal to
  - > Greater than
  - >= Greater than or equal to
  - < Less than
  - <= Less than or equal to
  - <> Not equal to
  - != Not equal to
- Boolean Operators
  - AND, OR, and NOT Operators for defining conditions in WHERE clause
  - Boolean Operators with Parenthesis can override normal precedence rules. In this case parentheses make the OR take place before the AND.

Variables and Parameters

- Must have a name that starts with '@'
- Used to:
  - temporarily store values that need to be reused across multiple queries.
  - Hold parameter values that are passed from an application
- Example:

```
Declare @myDate date = getdate()
Select orders from orderDetails
where orderDate < @mydate
```

Conditional Expressions – CASE (TSQL p.52-53)

A CASE expression acts like an if-then statement. It allows you to choose what will appear in a column of the result set, depending on a condition.

Conditional IF (TSQL p.367-368)

- Uses Boolean logic
- Example:

```
DECLARE @mydate DATE = '01-01-2010';
IF  (@mydate > '05/01/2015')
    PRINT 'date is more';
ELSE
    PRINT 'Date is less';
```

Query Multiple Tables - Joins

Joins: a relational operation that causes two or more tables with a common domain to be combined into a single table or view

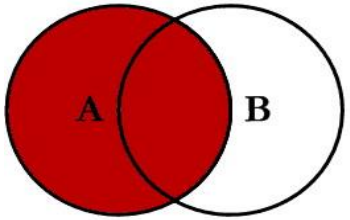
- Inner join
  - A join in which the joining condition is based on equality between values in the common columns; one of the duplicate columns is eliminated in the result table

- Outer join
  - A join in which rows that do not have matching values in common columns are nonetheless included in the result table (as opposed to **inner** join, in which rows must have matching values in order to appear in the result table)
  - Outer join can be used when we want to keep all the tuples of one, or both tables, in the result of the JOIN, regardless of whether or not they have matching tuples in the other table

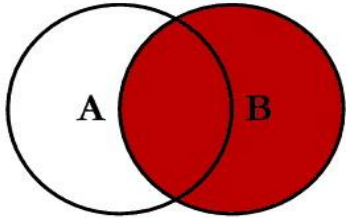
- Full join
  - Includes all data from each table that was joined

- Cross Join (Tsql p 103- 107 – hint on page 106 for one question in hw3)
  - Implements a cartesian product where each row is matched with all rows from the other table

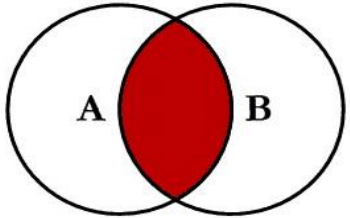
SQL JOINS



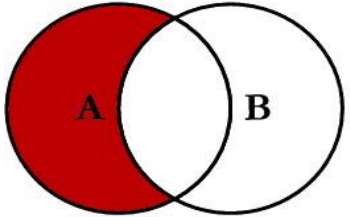

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```



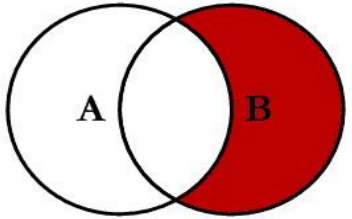

```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
```



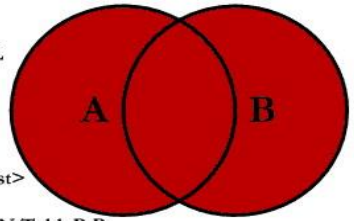

```
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
```



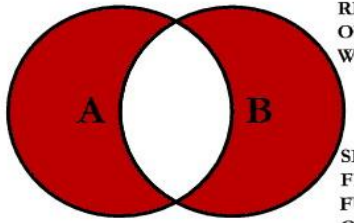

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
```




```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
```




```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
```




```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
OR B.Key IS NULL
```

Self Join

What are the employee ID and name of each employee and the name of his or her supervisor (label the supervisor's name Manager)?

The same table is used on both sides of the join; distinguished using table aliases.

Self-Join Query – TSQL4

```
Select e1.empid, e1.lastname, e1.city,e2.empid as
mgrid, e2.lastname as mgrLastName, e2.city as
mgrCity

from hr.Employees e1

inner join hr.Employees e2 on e1.mgrid = e2.empid
```

	empid	lastname	city	mgrid	mgrLastName	mgrCity
1	2	Funk	Tacoma	1	Davis	Seattle
2	3	Lew	Kirkland	2	Funk	Tacoma
3	4	Peled	Redmond	3	Lew	Kirkland
4	5	Mortensen	London	2	Funk	Tacoma
5	6	Suurs	London	5	Mortensen	London
6	7	King	London	5	Mortensen	London
7	8	Cameron	Seattle	3	Lew	Kirkland
8	9	Doyle	London	5	Mortensen	London

In class activity

Log into PMDB Playground to run DML

https://www.pdbmbook.com/practice

Simple Queries

- SQL statements that retrieve data from only one table

```
Q1:
SELECT SUPNR, SUPNAME,SUPADDRESS, SUPCITY, SUPSTATUS FROM SUPPLIER
Q1: SELECT * FROM SUPPLIER
```

SUPNR	SUPNAME	SUPADDRESS	SUPCITY	SUPSTATUS
21	Deliwines	240, Avenue of the Americas	New York	20
32	Best Wines	660, Market Street	San Francisco	90
37	Ad Fundum	82, Wacker Drive	Chicago	95
52	Spirits & co.	928, Strip	Las Vegas	NULL
68	The Wine Depot	132, Montgomery Street	San Francisco	10
69	Vinos del Mundo	4, Collins Avenue	Miami	92

Simple Queries

```
Q2: SELECT SUPNR,
SUPNAME FROM SUPPLIER
```

SUPNR	SUPNAME
21	Deliwines
32	Best Wines
37	Ad Fundum
52	Spirits & co.
68	The Wine Depot
69	Vinos del Mundo

```
Q3: SELECT SUPNR
    FROM PURCHASE_ORDER
```

SUPNR
32
32
37
37
37
37
37
68
69
94

Simple Queries

```
Q4: SELECT
DISTINCT SUPNR
FROM
PURCHASE_ORDER
```

SUPNR
32
37
68
69
94

```
Q5: SELECT SUPNR, PRODNR,
DELIV_PERIOD/30 AS
MONTH_DELIV_PERIOD FROM
SUPPLIES
```

SUPNR	PRODNR	MONTH_DELIV_PERIOD
21	0119	0.0333
21	0178	NULL
21	0289	0.0333
21	0327	0.2000
21	0347	0.0667
21	0384	0.0667
…	…	…

Simple Queries

```
Q6: SELECT SUPNR, SUPNAME FROM SUPPLIER
  WHERE SUPCITY = 'San Francisco'
```

SUPNR	SUPNAME	SUPSTATUS
32	Best Wines	90
68	The Wine Depot	10

```
Q7: SELECT SUPNR, SUPNAME
FROM SUPPLIER
WHERE SUPCITY = 'San
Francisco' AND SUPSTATUS > 80
```

SUPNR	SUPNAME	SUPSTATUS
32	Best Wines	90

Simple Queries

```
Q8: SELECT SUPNR, SUPNAME, SUPSTATUS
FROM SUPPLIER WHERE SUPSTATUS BETWEEN 70 AND 80
```

SUPNR	SUPNAME	SUPSTATUS
94	The Wine Crate	75

```
Q9:SELECT PRODNR, PRODNAME
   FROM PRODUCT
   WHERE PRODTYPE IN ('WHITE',
'SPARKLING')
```

PRODNR	PRODNAME
0178	Meerdael, Methode Traditionnelle Chardonnay, 2014
0199	Jacques Selosse, Brut Initial, 2012
0212	Billecart-Salmon, Brut Réserve, 2014
0300	Chateau des Rontets, Chardonnay, Birbettes
0494	Veuve-Cliquot, Brut, 2012
0632	Meneghetti, Chardonnay, 2010
…	….

Simple Queries

```
Q10: SELECT PRODNR, PRODNAME
     FROM PRODUCT
     WHERE PRODNAME LIKE '%CHARD%'
```

Note: underscore (_) is a substitute for a single character!

PRODNR	PRODNAME
0300	Chateau des Rontets, Chardonnay, Birbettes
0783	Clos D'Opleeuw, Chardonnay, 2012
0178	Meerdael, Methode Traditionnelle Chardonnay, 2014
0632	Meneghetti, Chardonnay, 2010

```
Q11: SELECT SUPNR, SUPNAME, SUPSTATUS
     FROM SUPPLIER
     WHERE SUPSTATUS IS NULL
```

SUPNR	SUPNAME	SUPSTATUS
52	Spirits & Co.	NULL

Queries with Aggregate Functions

```
Q12: SELECT COUNT(*)
     FROM SUPPLIES
     WHERE PRODNR = '0178'
```

5

```
Q13: SELECT COUNT(PURCHASE_PRICE)
     FROM SUPPLIES
     WHERE PRODNR = '0178'
```

4

```
Q14: SELECT COUNT(DISTINCT PURCHASE_PRICE)
     FROM SUPPLIES
     WHERE PRODNR = '0178'
```

3

Queries with Aggregate Functions

```
Q15: SELECT PRODNR, SUM(QUANTITY) AS
     SUM_ORDERS FROM PO_LINE
     WHERE PRODNR = '0178'
```

PONR	PRODNR	QUANTITY
…		
1512	0178	3
1538	0178	6
…		

0178 9

```
Q16: SELECT SUM(QUANTITY) AS
     TOTAL_ORDERS FROM PO_LINE
```

173

Queries with Aggregate Functions

```
Q17: SELECT PRODNR, AVG(PURCHASE_PRICE) AS
     WEIGHTED_AVG_PRICE FROM SUPPLIES
     WHERE PRODNR = '0178'
```

SUPNR	***PRODNR***	PURCHASE_PRICE	DELIV_PERIOD
…			
21	0178	NULL	NULL
37	0178	16.99	4
68	0178	17.99	5
69	0178	16.99	NULL
94	0178	18.00	6
…			

0178, (16.99+17.99+16.99+18.00)/4 = 17.4925

Queries with Aggregate Functions

```
Q18: SELECT PRODNR, AVG(DISTINCT
     PURCHASE_PRICE) AS UNWEIGHTED_AVG_PRICE
     FROM SUPPLIES WHERE PRODNR = '0178'
```

SUPNR	***PRODNR***	**PURCHASE_PRICE**	**DELIV_PERIOD**
…			
21	0178	NULL	NULL
37	0178	16.99	4
68	0178	17.99	5
69	0178	16.99	NULL
94	0178	18.00	6
…			

0178, (16.99+17.99+18.00)/3 = 17.66

Queries with Aggregate Functions

- **Q19:**
```
SELECT PRODNR, VARIANCE(PURCHASE_PRICE)
AS PRICE_VARIANCE FROM SUPPLIES
WHERE PRODNR = '0178'
```

PRODNR	PRICE_VARIANCE
0178	0.25251875000000024

Q20:
```
SELECT PRODNR, MIN(PURCHASE_PRICE) AS LOWEST_PRICE,
MAX(PURCHASE_PRICE) AS HIGHEST_PRICE
FROM SUPPLIES
WHERE PRODNR = '0178'
```

PRODNR	LOWEST_PRICE	HIGHEST_PRICE
0178	16.99	18.00

Queries with GROUP BY/HAVING

```
Q21: SELECT PRODNR
     FROM PO_LINE
     GROUP BY PRODNR
     HAVING COUNT(*) >= 3
```

PONR	PRODNR	QUANTITY
1511	0212	2
1512	0178	3
1513	0668	7
1514	0185	2
1514	0900	2
1523	0900	3
1538	0178	6
1538	0212	15
1560	0900	9
1577	0212	6
1577	0668	9
…	..	…

Queries with GROUP BY/HAVING

```
Q22: SELECT PRODNR, SUM(QUANTITY) AS QUANTITY FROM PO_LINE
     GROUP BY PRODNR
     HAVING SUM(QUANTITY) > 15
```

GROUP BY

PONR	PRODNR	QUANTITY
1511	0212	2
1577	0212	6
1538	0212	15
	SUM	23

PONR	PRODNR	QUANTITY
1512	0178	3
1538	0178	6
	SUM	9

PONR	PRODNR	QUANTITY
1513	0668	7
1577	0668	9
	SUM	16

PONR	PRODNR	QUANTITY
1514	0185	2
	SUM	2

PONR	PRODNR	QUANTITY
1514	0900	2
1523	0900	3
1560	0900	9
	SUM	14

PRODNR	QUANTITY
0212	23
0668	16

Queries with ORDER BY

```
Q23: SELECT PONR, PODATE, SUPNR
     FROM PURCHASE_ORDER
     ORDER BY PODATE ASC, SUPNR
DESC
```

PONR	PODATE	SUPNR
1511	2015-03-24	37
1512	2015-04-10	94
1513	2015-04-11	37
1514	2015-04-12	32
…		

```
Q24: SELECT PRODNR, SUPNR, PURCHASE_PRICE
     FROM SUPPLIES
     WHERE PRODNR = '0178'
     ORDER BY 3 DESC
```

PRODNR	SUPNR	PURCHASE_PRICE
0178	94	18.00
0178	68	17.99
0178	37	16.99
0178	69	16.99
0178	21	NULL

Inner Joins

SUPPLIER(SUPNR, SUPNAME, ..., SUPSTATUS)
SUPPLIES(*SUPNR*, *PRODNR*, PURCHASE_PRICE, ...)

SUPNR	SUPNAME	SUPADDRESS	SUPCITY	SUPSTATUS
32	Best wines			90
68	The Wine Depot			10
84	Wine Trade Logistics			92
:	:			:

SUPNR	*PRODNR*	PURCHASE_PRICE	DELIV_PERIOD
32	0474	40.00	1
32	0154	21.00	4
84	0494	15.99	2
:	:	:	

Inner Joins

```
Q25: SELECT R.SUPNR, R.SUPNAME, R.SUPSTATUS,
     S.SUPNR, S.PRODNR, S.PURCHASE_PRICE
     FROM SUPPLIER R, SUPPLIES S
```

R.SUPNR	R.SUPNAME	R.SUPSTATUS	S.SUPNR	S.PRODNR	S.PURCHASE_PRICE
21	Deliwines	20	21	0119	15.99
32	Best Wines	90	21	0119	15.99
37	Ad Fundum	95	21	0119	15.99
52	Spirits & co.	NULL	21	0119	15.99
…					
32	Best Wines	90	32	0154	21.00
37	Ad Fundum	95	32	0154	21.00
52	Spirits & co.	NULL	32	0154	21.00
…					
69	Vinos del Mundo	92	94	0899	15.00
84	Wine Trade Logistics	92	94	0899	15.00
94	The Wine Crate	75	94	0899	15.00

Inner Joins

Q26:

```
SELECT R.SUPNR, R.SUPNAME,
R.SUPSTATUS, S.PRODNR,
S.PURCHASE_PRICE
FROM SUPPLIER R, SUPPLIES S
WHERE R.SUPNR = S.SUPNR
```

R.SUPNR	R.SUPNAME	R.SUPSTATUS	S.SUPNR	S.PRODNR	S.PURCHASE_PRICE
21	Deliwines	20	21	0119	15.99
21	Deliwines	20	21	0178	NULL
21	Deliwines	20	21	0289	17.99
21	Deliwines	20	21	0327	56.00
21	Deliwines	20	21	0347	16.00
21	Deliwines	20	21	0384	55.00
21	Deliwines	20	21	0386	58.99
21	Deliwines	20	21	0468	14.99
21	Deliwines	20	21	0668	6.00
32	Best Wines	90	32	0154	21.00
32	Best Wines	90	32	0474	40.00
32	Best Wines	90	32	0494	15.00
32	Best Wines	90	32	0657	44.99
32	Best Wines	90	32	0760	52.00
…					

Inner Joins

```
Q27: SELECT R.SUPNR, R.SUPNAME, R.SUPSTATUS,
     S.PRODNR, S.PURCHASE_PRICE
     FROM SUPPLIER AS R INNER JOIN SUPPLIES AS S
     ON (R.SUPNR = S.SUPNR)
```

Inner Joins

```
Q28: SELECT R.SUPNR, R.SUPNAME, PO.PONR, PO.PODATE,
     P.PRODNR,P.PRODNAME, POL.QUANTITY
     FROM SUPPLIER R, PURCHASE_ORDER PO, PO_LINE
     POL, PRODUCT P
     WHERE (R.SUPNR = PO.SUPNR)
     AND (PO.PONR = POL.PONR)
     AND (POL.PRODNR = P.PRODNR)
```

R.SUPNR	R.SUPNAME	PO.PONR	PO.PODATE	P.PRODNR	P.PRODNAME	POL.QUANTITY
37	Ad Fundum	1511	2015-03-24	0212	Billecart-Salmon, Brut Réserve, 2014	2
37	Ad Fundum	1511	2015-03-24	0345	Vascosassetti, Brunello di Montalcino, 2004	4
37	Ad Fundum	1511	2015-03-24	0783	Clos D'Opleeuw, Chardonnay, 2012	1
37	Ad Fundum	1511	2015-03-24	0856	Domaine Chandon de Briailles, Savigny-Les-Beaune, 2006	9
94	The Wine Crate	1512	2015-04-10	0178	Meerdael, Methode Traditionnelle Chardonnay, 2014	3
…						

Inner Joins

```
Q29: SELECT R1.SUPNAME, R2.SUPNAME,
     R1.SUPCITY
     FROM SUPPLIER R1, SUPPLIER R2
     WHERE R1.SUPCITY = R2.SUPCITY
     AND (R1.SUPNR < R2.SUPNR)
```

SUPNAME	SUPNAME	SUPCITY
Best Wines	The Wine Depot	San Francisco

SUPNR	SUPNAME	SUPADDRESS	SUPCITY	SUPSTATUS
21	Deliwines	240, Avenue of the Americas	New York	20
32	Best Wines	660, Market Street	San Francisco	90
37	Ad Fundum	82, Wacker Drive	Chicago	95
52	Spirits & co.	928, Strip	Las Vegas	NULL
68	The Wine Depot	132, Montgomery Street	San Francisco	10
69	Vinos del Mundo	4, Collins Avenue	Miami	92

Inner Joins

```
Q30: SELECT R.SUPNAME
     FROM SUPPLIER R, SUPPLIES S
     WHERE R.SUPNR = S.SUPNR
     AND S.PRODNR = '0899'
```

Wine Crate

Inner Joins

```
Q31: SELECT DISTINCT R.SUPNAME
     FROM SUPPLIER R, SUPPLIES S, PRODUCT P
     WHERE S.SUPNR = R.SUPNR
     AND S.PRODNR = P.PRODNR
     AND P.PRODTYPE = 'ROSE'
```

SUPNAME
DeliWines
DeliWines
DeliWines
The Wine Depot

SUPNAME
DeliWines
The Wine Depot

Inner Joins

```
Q32: SELECT P.PRODNR, P.PRODNAME, SUM(POL.QUANTITY)
     FROM PRODUCT P, PO_LINE POL
     WHERE P.PRODNR = POL.PRODNR
     GROUP BY P.PRODNR
```

PRODNR	PRODNAME	SUM(POL.QUANTITY)
0178	Meerdael, Methode Traditionnelle Chardonnay, 2014	9
0185	Chateau Petrus, 1975	2
0212	Billecart-Salmon, Brut Réserve, 2014	23
0295	Chateau Pape Clement, Pessac-Léognan, 2001	9
0306	Chateau Coupe Roses, Granaxa, 2011	11
…		

Outer Joins

SUPNR	SUPNAME	SUPSTATUS	PRODNR	PURCHASE_PRICE
21	Deliwines	20	0119	15.99
21	Deliwines	20	0178	NULL
…				
37	Ad Fundum	95	0795	20.99
52	**Spirits & Co.**	**NULL**	**NULL**	**NULL**
68	The Wine Depot	10	0178	17.99
…				

Outer Joins

```
Q33: SELECT R.SUPNR, R.SUPNAME, R.SUPSTATUS,
     S.PRODNR, S.PURCHASE_PRICE
     FROM SUPPLIER AS R LEFT OUTER JOIN SUPPLIES AS S
     ON (R.SUPNR = S.SUPNR)
```

SUPNR	SUPNAME	SUPADDRESS	SUPCITY	SUPSTATUS
68	The Wine Depot			
21	Deliwines			
94	The Wine Crate			
..				

SUPNR	PRODNR	PURCHASE_PRICE	DELIV_PERIOD
21	0119	15.99	1
21	0289	17.99	1
68	0178	17.99	5
:	:	:	:

Outer Joins

```
Q34: SELECT P.PRODNR, P.PRODNAME, SUM(POL.QUANTITY)
     AS SUM FROM PO_LINE AS POL RIGHT OUTER JOIN PRODUCT AS P
     ON (POL.PRODNR = P.PRODNR)
     GROUP BY P.PRODNR
```

P.PRODNR	P.PRODNAME	SUM
0119	Chateau Miraval, Cotes de Provence Rose, 2015	NULL
0154	Chateau Haut Brion, 2008	NULL
0178	Meerdael, Methode Traditionnelle Chardonnay, 2014	9
0185	Chateau Petrus, 1975	2
0199	Jacques Selosse, Brut Initial, 2012	NULL
0212	Billecart-Salmon, Brut Réserve, 2014	23
…		