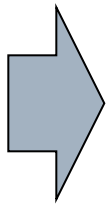
SQL: Data Manipulation

SQL Language Overview

SELECT Statement

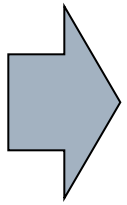
SQL Command Set – Core Commands

- Data Manipulation Language **(DML)**
  - SELECT…
  - INSERT…
  - UPDATE…
  - DELETE…



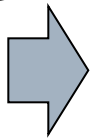
Actual Data

- Data Definition Language **(DDL)**
  - CREATE TABLE…
  - ALTER TABLE…
  - DROP TABLE…



Database Structure
(Tables, Views, Indexes)

- Data Control Language **(DCL)**
  - GRANT …
  - REVOKE…



Privileges

SELECT Statement

- Use the SELECT statement to retrieve data from a table.
- SELECT statement has 6 clauses:

SELECT	which columns to get
FROM	name of the table(s)
WHERE	which rows to get
GROUP BY	produce group summary data
HAVING	conditions for grouping
ORDER BY	how to sort the result

Only 'select' and 'from' clauses are mandatory

Select Syntax (simple format)

SELECT [DISTINCT|ALL]

    {* | [*column-expression* [AS *new-name*]] [, …]}

FROM *table-name* [*alias*][,…]

[WHERE *condition*]

[GROUP BY *column-list*]

[HAVING *condition*]

[ORDER BY *column-list*];

Note: terminating semi-colon is not Standard SQL, but most vendors require some sort of statement terminator.

SQL Coding Style

- Normally, we start each new clause on a new line, but this is not mandatory
- Some people use UPPER CASE for SQL reserved words and lower case for the names of data objects.

```
SELECT last_name, first_name
FROM l_employees
ORDER BY last_name;
```

SELECT example (Lunches DB)

```
SELECT employee_id, last_name, credit_limit
    FROM l_employees;
```

EMPLOYEE_ID	LAST_NAME	CREDIT_LIMIT
201	BROWN	30
202	KERN	25
203	WOODS	25
204	OWENS	15
205	PERKINS	25
206	ROSE	-
207	SMITH	25
208	CAMPBELL	25
210	HOFFMAN	25

More than 10 rows available. Increase rows selector to view more rows.

Notice the row limit message at the bottom of the data

SELECT example (Lunches DB)

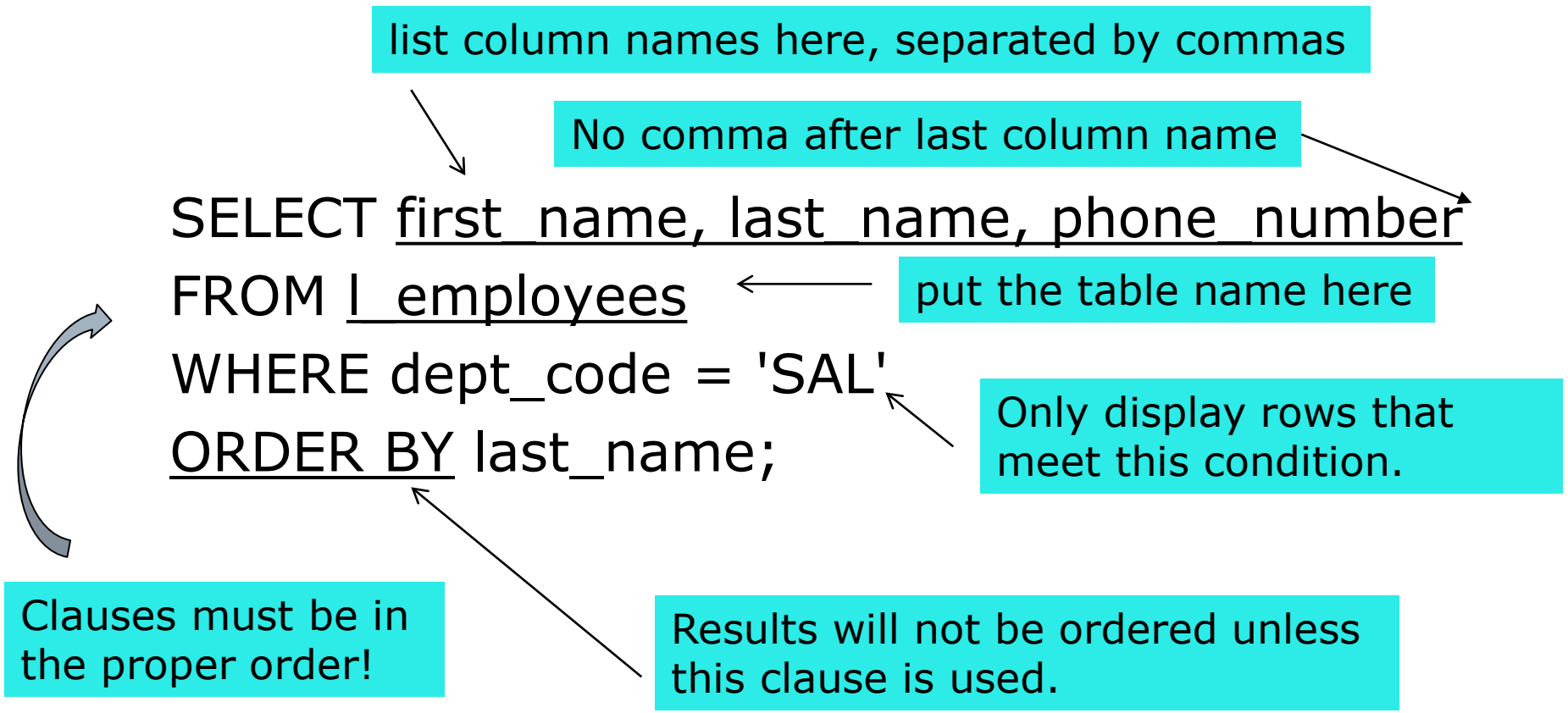
SELECT * FROM l_employees;

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	DEPT_CODE	HIRE_DATE	CREDIT_LIMIT	PHONE_NUMBER	MANAGER_ID
201	SUSAN	BROWN	EXE	01-JUN-98	30	3484	-
202	JIM	KERN	SAL	16-AUG-99	25	8722	201
203	MARTHA	WOODS	SHP	02-FEB-09	25	7591	201
204	ELLEN	OWENS	SAL	01-JUL-08	15	6830	202
205	HENRY	PERKINS	SAL	01-MAR-06	25	5286	202
206	CAROL	ROSE	ACT	-	-	-	-
207	DAN	SMITH	SHP	01-DEC-08	25	2259	203
208	FRED	CAMPBELL	SHP	01-APR-08	25	1752	203
210	NANCY	HOFFMAN	SAL	16-FEB-07	25	2974	203

More than 10 rows available. Increase rows selector to view more rows.

‘*’ Wildcard gives you a quick way to see all columns

SELECT Statement Details



More on SELECT

use any number of columns and in any order here

use 'AS' to display column name differently

```
SELECT phone number AS phone, last_name
FROM l_employees
WHERE dept_code = 'SAL'
ORDER BY last_name, first_name;
```

Can ORDER BY multiple columns

End your SQL statements with a semicolon

Use of Quotation Marks

```
SELECT first_name, last_name,
    phone_number, hire_date
FROM l_employees
WHERE last_name = 'SMITH' OR
          phone_number = 1752 OR
          hire_date > '1/1/2008';
```

Use single quotes around strings. This is the SQL standard!

DON'T use quotes around numbers.

Use single quotes around dates also.

Use of Double Quotes

- When renaming columns for nice output, use double quotes to surround the name of a column alias if it contains special characters or spaces.
  - SELECT hire_date AS "Hire Date", …
  - SELECT ft_sec AS "Feet/Second", …

IMPORTANT: Plain text editors reliably produce a straight single quote; If you cut & paste from WORD or PowerPoint, you might not get straight quotes. SQL processors will give you an error!

Literals in other products

- MySQL, SQL Server,and Access all support use of single or double quotes for delimiting character and date strings.
- Most of these products also have a configuration option that will enforce the stricter standard mode of using single quotes.
  - QUOTED_IDENTIFIER (Microsoft)
  - ANSI_QUOTES (MySQL)
- **For portability, use single quotes!**
- PostgreSQL uses double quotes for quoting reserved words

SELECT – Renaming Columns

- A column can be displayed with a different name by giving it an alias.

Format:

SELECT *column_name* **AS *alias_name***

Example:

```
SELECT
employee_id AS "employee number",
last_name AS "last name"
FROM l_employees;
```

Output:

```
EMPLOYEE NUMBER      LAST NAME
-------------------------------------
             201     BROWN
             202     KERN              (output truncated)
```

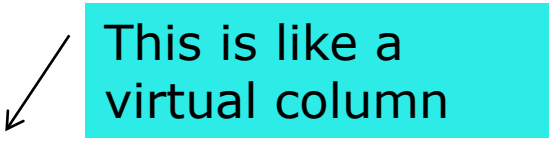
The alias name appears only in the output, you are not changing the column name inside the database!

SELECT Statement – Literals

- You can add a literal to the SELECT clause. It will appear in every column

```
SELECT employee_id, last name, 'EXCELLENT' AS rating
FROM l_employees;
```

Output would be:



EMPLOYEE_ID	LAST_NAME	RATING
201	BROWN	EXCELLENT
202	KERN	EXCELLENT
203	WOODS	EXCELLENT
204	OWENS	EXCELLENT

SELECT Statement - DISTINCT

- Use the DISTINCT clause to eliminate duplicates from the result table

```
SELECT DISTINCT manufacturer
FROM notebook_systems;
```

- Distinct clause can be used on more than 1 column

```
SELECT DISTINCT manufacturer, model
FROM notebook_systems;
```

Now both columns together must be distinct.

SELECT DISTINCT

Manufacturer	Model
HP	550
Acer	2983
Gateway	2200
HP	2010
Acer	2983
HP	550
Gateway	2800

Given this data, what are the results of:

- SELECT DISTINCT manufacturer
- SELECT DISTINCT manufacturer, model

SELECT with calculated fields

```
SELECT StaffNo, Fname, Lname, Salary/12
FROM Staff;
```

- Salary/12 is a derived or calculated field.
- You can use simple or complex expressions to create a derived field.
- Can use +, -, /, *, parentheses, etc. in calculations
- Nice to give the derived column an alias name:
  SELECT …, Salary/12 AS "Monthly Salary" ….

SELECT –
How many rows are returned?

```
SELECT * FROM l_employees;
```

!! BE CAREFUL !! In large DB, this will return **all rows**. You might get millions of rows returned and really slow things down. Only do this when you know the table is small.

Possible solutions:

→ Add a **WHERE clause** to restrict the number of rows returned.

→ Use **TOP/LIMIT/ROWNUM** to get first n records

→ Use **SAMPLE** in Standard SQL

Using Top/Limit/Rownum

- These commands will display the first n records in a table (after any ORDER BY has taken effect).
- In Microsoft SQL Server:

```
SELECT TOP 5 * FROM l_employees;
```

- In PostgreSQL and MySQL:

```
SELECT * FROM l_employees LIMIT 5;
```

- In Oracle:

```
SELECT * FROM l_employees
WHERE ROWNUM <=5;
```

Using TableSample

- TableSample will randomly select a certain percentage of the records.
- Use it to get a quick idea of data values, range
- Oracle:

```
SELECT * FROM l_lunch_items  SAMPLE (5);
```

- MySQL: *(see link below for working with larger tables)*

```
SELECT * FROM l_lunch_items
ORDER BY RAND() LIMIT 10
```

- PostgreSQL and MS SQL Server:

```
SELECT * FROM l_lunch_items
TABLESAMPLE SYSTEM (5);
```

http://jan.kneschke.de/projects/mysql/order-by-rand/

Filtering Data: WHERE clause

- Use the WHERE clause to be very specific about restricting the number of rows in the result table
- Five basic search conditions:
  1. Comparison
  2. Range
  3. Set membership
  4. Pattern match
  5. Null

WHERE CLAWS!!!

1. Comparison Search Condition

```
SELECT StafffNo, Fname, Lname, position, salary
FROM Staff
WHERE salary > 10000;
```

Comparison operators in SQL

=	equals
<> , !=	not equal (<> is ANSI Standard)
<, <=	less than, less than or equal
>, >=	greater than, greater than or equal
IN, NOT IN	set membership
BETWEEN, NOT BETWEEN	range comparison
LIKE, NOT LIKE	pattern match
IS NULL, IS NOT NULL	null test

Don't use = when comparing to NULL!

Unlike Java, PHP & other languages, SQL does not use == for comparisons.

2. Range Condition
BETWEEN, NOT BETWEEN

```
SELECT last_name, job_id, salary
FROM employees
WHERE salary BETWEEN 1000 AND 3000;
```

LAST_NAME	JOB_ID	SALARY
Baida	PU_CLERK	2900
Tobias	PU_CLERK	2800
Himuro	PU_CLERK	2600
Colmenares	PU_CLERK	2500
Mikkilineni	ST_CLERK	2700
Landry	ST_CLERK	2400

Note: BETWEEN includes the endpoints of the range; salary of 1000 and 3000 will be included.

BETWEEN with characters

- If you would like to SELECT a range of rows based upon the first letter of a name, be careful with the limits you specify:

```
SELECT * FROM l_employees
WHERE first_name BETWEEN 'A' and 'C';
```

will not return any names beginning with 'C'

- Instead you need to do this:

```
SELECT * FROM l_employees
WHERE first_name BETWEEN 'A' and 'D';
```

BETWEEN with dates

```
SELECT * FROM orders
WHERE date BETWEEN
    '1/1/2015' AND '2/28/2015';
```

- Be careful with range checking dates. SQL dates include date and time, but time is not always displayed or specified in the BETWEEN clause.
- This query will NOT return any orders from 2/28/2015 because there is an **implied time of 12:00 am** appended to 2/28/2015 and all values in the database would have a time greater than 12:00 AM
- This query will give us the results we want.

```
SELECT * FROM orders
WHERE date BETWEEN
    '1/1/2015' AND '3/1/2015';
```

A slight diversion…
Default Date Formats

- **Oracle-** default date format is: DD-MON-YYYY or DD-MON-YY. Example: '13-JAN-2016'. Use this with SQL*Plus.
- **Oracle – Application Express-** default date format is: mm/dd/yyyy. Example: '1/13/2016'
- **PostgreSQL and MySQL-** default date format is 'yyyy-mm-dd'. Example: '2016-01-13'

Date Format Conversions

- These can be tricky! There is no standard way.
- Just ask yourself….Is it input or output?
  - Are you changing the format for ouput? (e.g. display?)
  - Are you trying to insert data into the database that is not in the default format?
- When these are true, you need a date conversion function. Often, we Use different functions for input and output.

Oracle
Date Format Conversions

- Whenever you want to output (display) a date in a non-default format, use the **to_char()** function:

```
SELECT to_char(hire_date, 'YYYY-MM-DD')
FROM l_employees;
```

- Whenever you want to insert a date that is not in the default format, use the **to_date()** function and specify the date format in the 2nd parameter.

```
INSERT into l_employees (last_name, hire_date)
VALUES ('Johnson', to_date('2013-01-07', 'yyyy-mm-dd');
```

Oracle
Date Format Conversions

- In a WHERE clause, use the **to_date()** function

```
SELECT * FROM l_employees
WHERE hire_date =
TO_DATE('1998/06/01','YYYY/MM/DD');
```

- Here is the same statement using Oracle Application Express default date format:

```
SELECT * FROM l_employees
WHERE hire_date = '6/1/1998';
```

MySQL
Date Format Conversions

- To change the displayed format of a date, use **date_format()** function:

```
SELECT date_format (lunch_date, "%m/%d/%Y");
FROM l_lunches;
```

OUTPUTS dates in this format: 11/30/2016

- To insert a date that is not in the default format, use **str_to_date()** function:

```
INSERT into l_employees (last_name, hire_date)
VALUES('Johnson',str_to_date('5/6/2016', '%m/%d/%Y'));
```

Format of inserted date is: 2016-05-06

http://www.w3resource.com/mysql/date-and-time-functions/mysql-date_format-function.php

MySQL
Date Format Conversions

- In a WHERE clause, use the **str_to_date()** function

```
SELECT * FROM l_employees
      WHERE hire_date =
   STR_TO_DATE ('06/01/1998','%m/%d/%Y');
```

- Here is the same statement using MySQL default date format:

```
SELECT * FROM l_employees
WHERE hire_date = '1998-06-01';
```

3. Set Membership – IN/NOT IN

```
SELECT StaffNo, Fname, Lname, position
FROM staff
WHERE position IN ('Manager', 'Supervisor');
```

Notes: Can also do this in SQL without the IN operator:

```
SELECT StaffNo, Fname, Lname, position
FROM staff
WHERE position = 'Manager' OR
        position = 'Supervisor';
```

Tip: Many students forget that they can use IN clause. It's really much shorter and shows a better understanding of SQL.

Why is 'IN' operator so useful?

- With just one or two items it doesn't matter whether you use IN or '=' operator.
- However, with lots of items in the set, IN is very convenient.

```
SELECT * from employees
WHERE dept_code IN ('SAL', 'MKT', 'ACT', 'IT', 'MFG', 'SVC');
```

- This statement condenses 6 tests for equality into one test. Much simpler to code!

4. Pattern matching with 'LIKE'

- Wildcard characters
  - **% - matches a string of 0 or more characters**
  - _ (underscore) – matches 1 character
- Example,

```
SELECT last_name, first_name FROM l_employees
WHERE last_name LIKE 'B%';
```

LAST_NAME	FIRST_NAME
BUTTON	BENJAMIN
BELL	LULU
BROWN	SUSAN

- Escape a wildcard with the backslash: \%, _

4. Pattern matching with 'LIKE'

- Wildcard characters
  - % - matches a string of 0 or more characters
  - **_ (underscore) – matches 1 character**
- Example,

  SELECT last_name, first_name FROM l_employees
  WHERE first_name like '_A%'

LAST_NAME	FIRST_NAME
ROSE	CAROL
SMITH	DAN
WOODS	MARTHA
HOFFMAN	NANCY
TINKER	SAM

- Escape a wildcard with the backslash: \%, _

Regular Expressions

- Very powerful string matching functions. They are commonly used in most programming languages and scripts like sed, awk, Perl.
- Most database vendors support the use of regular expressions with special SQL "LIKE" operator.
- "LIKE" is the little hammer and "REGEX" is the big hammer. Don't use the "bigger hammer" unless you need it.
- Save Regular Expressions for when LIKE will not work.

REGEXP	Action
*	Matches zero or more instances of the string preceding it
+	Matches one or more instances of the string preceding it
?	Matches zero or one instances of the string preceding it
.	Matches any single character, except a newline
[xyz]	Matches any of x, y, or z (match one of enclosed chars)
[A-Z] [a-z] [0-9]	Matches any uppercase letter Matches any lowercase letter Matches any digit
^ $	Anchors the match from the beginning Anchors the match to the end
{n} {n,m}	String must occur exactly n times String may occur n(min) to m(max) times
|	(OR) Separates alternatives

Note: There are MANY more operations and actions! Use online references for more details.

Regular Expressions in Oracle:

- Oracle uses the REGEXP_LIKE function to add the power of regular expressions to it's SQL:

```
SELECT last_name FROM l_employees
WHERE REGEXP_LIKE(last_name,'[M-Z]');
```

LAST_NAME
WOODS
OWENS
PERKINS
ROSE
SMITH
SPASYK
MILLER

- What does this regexp do?

```
SELECT id FROM employees
WHERE REGEXP_LIKE
(id,'^[0-9]{3}-[0-9]{2}-[0-9]{4}$');
```

Regular Expressions in MySQL

- Use 'REGEXP'
- Example,

```
SELECT *
FROM l_employees
WHERE last_name REGEXP '^S';
```

will select last names that begin with 'S'

More Information on Regular Expressions

- For General Information on Regular Expressions:

http://en.wikipedia.org/wiki/Regular_expression

- For Oracle Regular Expressions:

http://www.psoug.org/reference/regexp.html

http://www.databasejournal.com/features/oracle/article.php/3501826/Oracle-and-Regular-Expressions.htm

- For MySQL Regular Expressions:

http://www.go4expert.com/forums/showthread.php?t=2337

Standard Escape Sequence

- Each dialect of SQL may have it’s own escape sequence. Let’s say you want to search for a string that contains the underscore (‘_’). Can use standard method to escape the underscore:

```
LIKE 'ADAMS\_APPLE'  ESCAPE  '\'
```

- Oracle supports this ESCAPE clause
- If you want to ESCAPE a single quote, use TWO single quotes

```
WHERE title = 'Three''s Company'
```

5. NULL in Search Conditions

```
SELECT last_name, credit_limit
FROM l_employees
WHERE credit_limit IS NULL;
```

LAST_NAME	CREDIT_LIMIT
ROSE	-
BUTTON	-
BELL	-
TINKER	-
SCOTT	-
SCOTT	-
SMITH	-
MILLER	-

!! Do not use WHERE [col] = NULL. It won't work!

5. NOT NULL in Search Conditions

```
SELECT last_name, credit_limit
FROM l_employees
WHERE credit_limit IS NOT NULL;
```

LAST_NAME	CREDIT_LIMIT
BROWN	30
KERN	25
WOODS	25
OWENS	15
PERKINS	25
SMITH	25
CAMPBELL	25

!! Do not use WHERE [col] != NULL. It won't work!

Complex Comparisons

- Use logical operators AND, OR, NOT to create more complex expressions
- Order of evaluation is:
  - Left to right
  - Expressions in parentheses are evaluated first
  - NOTs are evaluated before ANDs, Ors
  - ANDs are evaluated before ORs

WHERE clause
Complex expressions

```
SELECT employee_id, dept_code, hire_date
FROM l_employees
WHERE NOT(dept_code IN ('SAL','SHP')
      OR employee_id BETWEEN 202 AND 205
      OR hire_date IS NOT NULL);
```

- WHERE expressions can often be VERY long. This is really a simple one!

WHERE clause – Negation

Oracle & MySQL allow:

1. SELECT * FROM l_employees
   WHERE manager_id **<>** 201;
2. SELECT * FROM l_employees
   WHERE **not** (manager_id = 201);
3. SELECT * FROM l_employees
   WHERE manager_id **!=** 201;

Only Oracle allows:

4. SELECT * FROM l_employees
   WHERE manager_id **^=** 201; -- not common

Order By Clause

- Use ORDER BY to sort the results.
- Can be in ascending or descending order. (ASC, DESC)
- Specify ASC|DESC for each column
- Ascending is the default
- Can sort on multiple columns
- Can specify column name or positional number
- In Oracle, nulls appear at bottom of sort order

Order by - Examples

- ORDER BY employee_id;
- ORDER BY last_name, first_name;
- ORDER BY employee_id desc;
- ORDER BY hire_date asc;
- ORDER BY dept_code, last_name desc;
- ORDER BY hire_date desc, last_name asc;
- ORDER BY 1; -- sort by first column
- ORDER BY 4, 1; -- sort by fourth, first column

ORDER By - multiple columns

```
SELECT dept_code AS Dept, credit_limit AS CL,
          last_name, first_name
FROM l_employees
ORDER BY dept_code asc, credit_limit desc, last_name ASC;
```

Dept	CL	last_name	first_name
ACT	-	ROSE	CAROL
EXE	30	BROWN	SUSAN
MKT	15	JACOBS	PAULA
SAL	25	HOFFMAN	NANCY
SAL	25	KERN	JIM
SAL	25	PERKINS	HENRY
SAL	15	OWENS	ELLEN
SHP	25	CAMPBELL	FRED
SHP	25	SMITH	DAN
SHP	25	WOODS	MARTHA

Notes:

2nd sort column is not used until values in 1st sort column are identical

3rd sort column is not used until values in 1st and 2nd sort columns are identical.

Quick Word about Case Sensitivity

- Most RDBMS vendors allow you to specify case sensitivity when you install the server.
- In Oracle XE, strings will be case sensitive
- In MySQL , strings will be case insensitive.
- The **data in lunches database will be stored in upper case.**
- You may type SQL commands in upper, lower, or mixed case.

Case Sensitivity: Bottom Line

- For the Lunches database:
  - **Oracle** users should always search for strings using **UPPERCASE.**
  - **MySQL** users can enter strings in **upper or lower case** and they will still get a match. I believe this is because the XAMPP MySQL Server has been configured to be case insensitive.

More Quick Tips

- BE CAREFUL with cut and paste from these slides or and Microsoft document. SQL requires straight quotes, not smart quotes.
- Remember that SQL uses single quotes most of the time.
- SQL Workshop only allows you to execute one command at a time unless you are executing a script. If you have multiple commands in the window, highlight just one command to run it.

We strive for SQL Standards

- 1986 – American National Standards Institute (ANSI) developed first SQL Standard.
- Updates to the standard have been published in:
  - 1989 – Addendum for integrity enhancement
  - 1992 - SQL2 (also called SQL-92)
  - 1999 - SQL-1999
  - 2003 - SQL-2003,
  - More updates in 2006, 2008, 2011
- Bottom line – lots of updates to the standards. A little difficult to know exactly what's supported when a vendor claims conformance with the standard.

Importance of the SQL Standard

- All sales of RDBMS to US government must conform to standards
- SQL is part of FIPS – Federal Information Processing Standard.
- Using standard SQL, to the largest extent possible, eases portability.
- Discussion: pros and cons of portability.