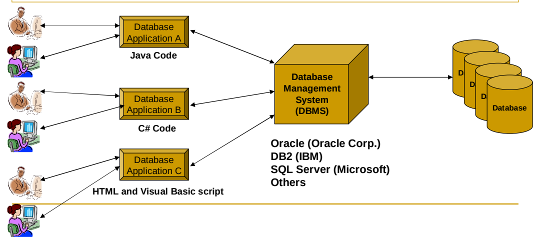
COP 3710: Database Management Systems

Basic SQL (PART ONE)

FAMU CIS Department
Ms. Chatmon

Relational Database Management System (RDBMS)

An RDBMS is the software program used to create the database and it allows you to enter, manipulate, and retrieve data



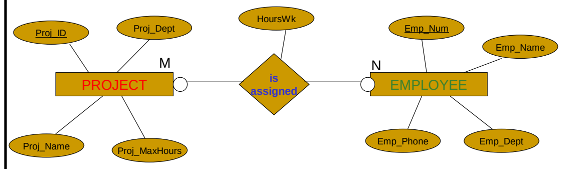
Sample Database: Case Study

- Business Rules
- E-R Diagram Level One
- E-R Diagram Level Two
- Tables w Sample Data

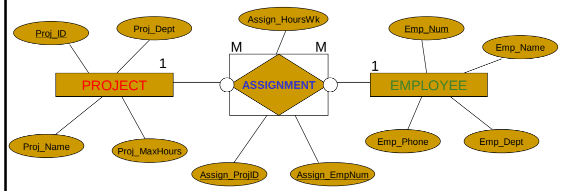
Case Study: Business Rules

- Each PROJECT is assigned to one or more EMPLOYEE(s), or none at all.
- Each EMPLOYEE is assigned to one or more PROJECT(s), or none at all.

Case Study: E-R Level One



Case Study: E-R Level Two



Case Study: Relations

- **PROJECT** (*Proj_Id*, Proj_Name, Proj_Dept, Proj_MaxHours)
- **EMPLOYEE** (*Emp_Num*, Emp_Name, Emp_Phone, Emp_Dept)
- **ASSIGNMENT** (*Assign_Empnum*, *Assign_ProjId*, Assign_HoursWk)

Case Study: Tables (Relations) w/ sample data

Table Name: PROJECT

Proj_ID	Proj_Name	Proj_Dept	Proj_MaxHours
1000	Q3 Portfolio Analysis	Finance	75.0
1200	Q3 Tax Prep	Accounting	145.0
1400	Q4 Product Plan	Marketing	138.0
1500	Q4 Portfolio Analysis	Finance	110.0

Table Name: ASSIGNMENT

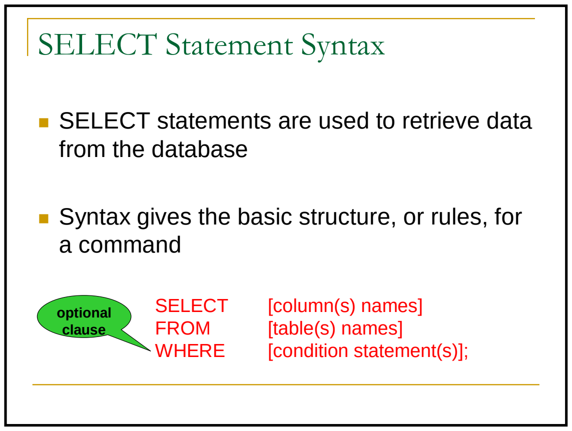
Assign_ProjID	Assign_EmpNum	Assign_HoursWk
1000	100	17.50
1000	300	12.50
1000	400	8.00
1000	500	20.25
1200	100	45.75
1200	400	70.50
1200	600	40.50
1400	200	75.00
1400	700	20.25
1400	500	25.25

Table Name: EMPLOYEE

Emp_Num	Emp_Name	Emp_Phone	Emp_Dept
100	Mary Jacobs	285-8879	Accounting
200	Kenji Numoto	287-0098	Marketing
300	Heather Jones	287-9981	Finance
400	Rosalie Jackson	285-1273	Accounting
500	James Nestor		Info Systems
600	Richard Wu	287-0123	Info Systems
700	Kim Sung	287-3222	Marketing

SQL for Relational Query

- Reading Specified Columns from single Table
- Reading Specified Rows from a single Table
- Reading Specified Columns and Specified Rows from a single Table
- Ranges, Wildcards, and Nulls in Where Clauses
- Sorting the Results
- SQL Built-In Functions

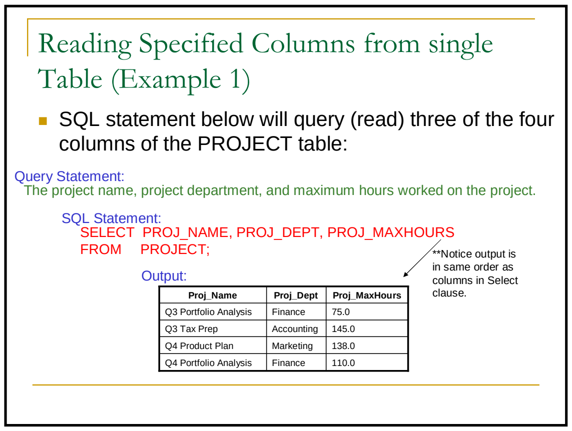


SELECT Statement Syntax

- SELECT statements are used to retrieve data from the database
- Syntax gives the basic structure, or rules, for a command

optional clause

```
SELECT     [column(s) names]
FROM       [table(s) names]
WHERE      [condition statement(s)];
```



Reading Specified Columns from single Table (Example 1)

- SQL statement below will query (read) three of the four columns of the PROJECT table:

Query Statement:
The project name, project department, and maximum hours worked on the project.

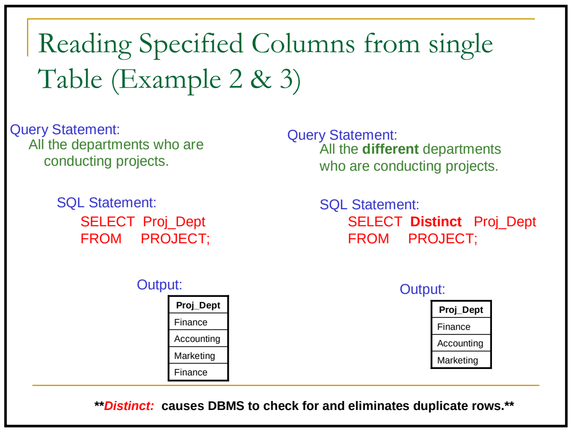
SQL Statement:

```
SELECT  PROJ_NAME, PROJ_DEPT, PROJ_MAXHOURS
FROM    PROJECT;
```

**Notice output is in same order as columns in Select clause.

Output:

Proj_Name	Proj_Dept	Proj_MaxHours
Q3 Portfolio Analysis	Finance	75.0
Q3 Tax Prep	Accounting	145.0
Q4 Product Plan	Marketing	138.0
Q4 Portfolio Analysis	Finance	110.0



Reading Specified Columns from single Table (Example 2 & 3)

Query Statement:
All the departments who are conducting projects.

SQL Statement:

```
SELECT  Proj_Dept
FROM    PROJECT;
```

Output:

Proj_Dept
Finance
Accounting
Marketing
Finance

Query Statement:
All the **different** departments who are conducting projects.

SQL Statement:

```
SELECT  Distinct  Proj_Dept
FROM    PROJECT;
```

Output:

Proj_Dept
Finance
Accounting
Marketing

****Distinct:* causes DBMS to check for and eliminates duplicate rows.****

Reading Specified Rows from single Table (Example 1)

- SQL statements below are used to select all the columns for certain rows.

Query Statement:
The project id, name, department, and maximum hours of those projects sponsored by the finance department.

SQL Statement:
```
SELECT    PROJ_ID, PROJ_NAME, PROJ_DEPT, PROJ_MAXHOURS
FROM      PROJECT
WHERE     PROJ_DEPT = 'Finance';
```

```
SELECT    *
FROM      PROJECT
WHERE     PROJ_DEPT = 'Finance';
```

Output:

Proj_ID	Proj_Name	Proj_Dept	Proj_MaxHours
1000	Q3 Portfolio Analysis	Finance	75.0
1500	Q4 Portfolio Analysis	Finance	110.0

Reading Specified Rows from single Table (Example 2)

Query Statement:
All project information relating to those projects sponsored by the finance department where the maximum hours worked on the project exceed 100 hours.

SQL Statement:
```
SELECT    *
FROM      PROJECT
WHERE     PROJ_DEPT = 'Finance' AND PROJ_MAXHOURS > 100;
```

Output:

Proj_ID	Proj_Name	Proj_Dept	Proj_MaxHours
1500	Q4 Portfolio Analysis	Finance	110.0

Reading Specified Columns and Specified Rows from a single Table

- SQL statements below are used to select some columns and some rows from a table.

Query Statement:
The name and department of employees in the Accounting department.

SQL Statement:
```
SELECT    EMP_NAME, EMP_DEPT
FROM      EMPLOYEE
WHERE     EMP_DEPT = 'Accounting';
```

Output:

Emp_Name	Emp_Dept
Mary Jacobs	Accounting
Rosalie Jackson	Accounting

Reading Specified Columns and Specified Rows from a single Table

- A column could have **one** of a set of values in a list, which can be defined using the **IN** operator.

Query Statement:

The name, phone, and department of employees in either the Accounting, Finance, or Marketing department.

SQL Statement:

```
SELECT   EMP_NAME, EMP_PHONE, EMP_DEPT
FROM     EMPLOYEE
WHERE    EMP_DEPT IN ('Accounting', 'Finance', 'Marketing');
```

Output:

Emp_Name	Emp_Phone	Emp_Dept
Mary Jacobs	285-8879	Accounting
Kenji Numoto	287-0098	Marketing
Heather Jones	287-9981	Finance
Rosalie Jackson	285-1273	Accounting
Kim Sung	287-3222	Marketing

Reading Specified Columns and Specified Rows from a single Table

Query Statement:

The name, phone, and department of employees not in the Accounting, Finance, or Marketing departments.

SQL Statement:

```
SELECT   EMP_NAME, EMP_PHONE, EMP_DEPT
FROM     EMPLOYEE
WHERE    EMP_DEPT  NOT  IN  ('Accounting', 'Finance', 'Marketing');
```

Output:

Emp_Name	Emp_Phone	Emp_Dept
James Nestor		Info Systems
Richard Wu	287-0123	Info Systems

Ranges, Wildcards, and Nulls in Where Clauses

- WHERE clause can also refer to ranges of values.

Query Statement:

The employee names and departments of those employees who have employee numbers ranging from 200 to 500.

SQL Statement (version #1):

```
SELECT  EMP_NAME, EMP_DEPT
FROM    EMPLOYEE
WHERE   EMP_NUM BETWEEN 200 AND 500;
```

SQL Statement (version #2):

```
SELECT  EMP_NAME, EMP_DEPT
FROM    EMPLOYEE
WHERE   EMP_NUM  =>  200
AND     EMP_NUM  =<  500;
```

Output:

Emp_Name	Emp_Dept
Kenji Numoto	Marketing
Heather Jones	Finance
Rosalie Jackson	Accounting
James Nestor	Info Systems

Ranges, Wildcards, and Nulls in Where Clauses

- WHERE clause can also refer to partial values.
  - The **LIKE** operator is used to select partial values.
  - The **underscore symbol** (_) represents a single, unspecified character. (pattern matching)

Query Statement:
All employees who have a phone number that begins with 285-.

SQL Statement:

four underscores

```
SELECT   *
FROM     EMPLOYEE
WHERE    EMP_PHONE  LIKE  '285-____';
```

Output:

100	Mary Jacobs	285-8879	Accounting
400	Rosalie Jackson	285-1273	Accounting

*******Underscore (_)***: any character can occur in that spot.****

Ranges, Wildcards, and Nulls in Where Clauses

- WHERE clause can also refer to partial values.
  - The **LIKE** operator is used to select partial values.
  - The **percent sign** (%) is used to represent a series of one or more unspecified characters.

Query Statement:
All employees who have a phone number that begins with 285-.

SQL Statement:

```
SELECT   *
FROM     EMPLOYEE
WHERE    EMP_PHONE  LIKE  '285-%';
```

Output:

100	Mary Jacobs	285-8879	Accounting
400	Rosalie Jackson	285-1273	Accounting

Ranges, Wildcards, and Nulls in Where Clauses

- The WHERE clause could include the keyword **IS NULL** to search for null values.

Query Statement:
The names and departments of all employees who don't have a phone number.

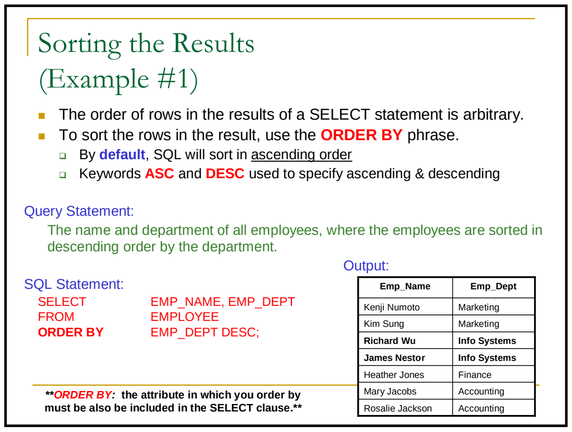
SQL Statement:

```
SELECT   EMP_NAME, EMP_DEPT
FROM     EMPLOYEE
WHERE    EMP_PHONE IS NULL;
```

Output:

James Nestor	Info Systems

*******IS NOT NULL***: this can be used to indicate that there is not the presence of a null value.****



Sorting the Results (Example #1)

- The order of rows in the results of a SELECT statement is arbitrary.
- To sort the rows in the result, use the **ORDER BY** phrase.
  - By **default**, SQL will sort in ascending order
  - Keywords **ASC** and **DESC** used to specify ascending & descending

Query Statement:
The name and department of all employees, where the employees are sorted in descending order by the department.

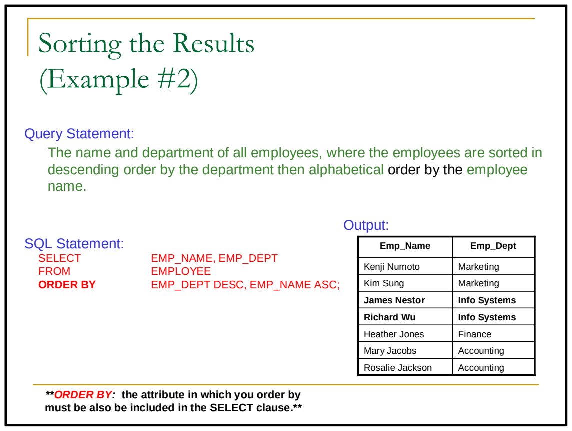
SQL Statement:

```
SELECT      EMP_NAME, EMP_DEPT
FROM        EMPLOYEE
ORDER BY    EMP_DEPT DESC;
```

Output:

Emp_Name	Emp_Dept
Kenji Numoto	Marketing
Kim Sung	Marketing
Richard Wu	**Info Systems**
James Nestor	**Info Systems**
Heather Jones	Finance
Mary Jacobs	Accounting
Rosalie Jackson	Accounting

****ORDER BY*: the attribute in which you order by must be also be included in the SELECT clause.****



Sorting the Results (Example #2)

Query Statement:
The name and department of all employees, where the employees are sorted in descending order by the department then alphabetical order by the employee name.

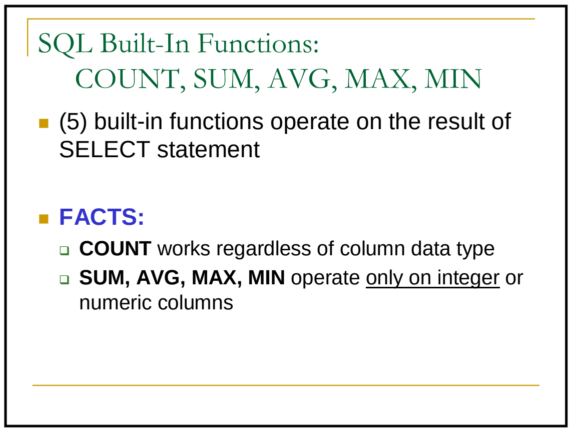
SQL Statement:

```
SELECT      EMP_NAME, EMP_DEPT
FROM        EMPLOYEE
ORDER BY    EMP_DEPT DESC, EMP_NAME ASC;
```

Output:

Emp_Name	Emp_Dept
Kenji Numoto	Marketing
Kim Sung	Marketing
James Nestor	**Info Systems**
Richard Wu	**Info Systems**
Heather Jones	Finance
Mary Jacobs	Accounting
Rosalie Jackson	Accounting

****ORDER BY*: the attribute in which you order by must be also be included in the SELECT clause.****



SQL Built-In Functions: COUNT, SUM, AVG, MAX, MIN

- (5) built-in functions operate on the result of SELECT statement
- **FACTS:**
  - **COUNT** works regardless of column data type
  - **SUM, AVG, MAX, MIN** operate only on integer or numeric columns

SQL Built-In Functions: COUNT

- COUNT function counts the number of rows in the result

Query Statement #1:
The number of current projects.

SQL Statement:
```
SELECT   COUNT(*)
FROM     PROJECT;
```
Output:
$\boxed{4}$

Query Statement #2:
The number of the departments who have projects going on.

SQL Statement:
```
SELECT   COUNT (PROJ_DEPT)
FROM     PROJECT;
```
Output:
$\boxed{4}$

Query Statement #3:
The number of the different departments who have projects going on.

SQL Statement:
```
SELECT   COUNT (DISTINCT  PROJ_DEPT)
FROM     PROJECT;
```
Output:
$\boxed{3}$

SQL Built-In Functions: SUM, MIN, MAX

- SUM totals the set of values of a numeric column; MIN finds the minimum value in the column; MAX finds the maximum value in the column;

SQL Statement:
```
SELECT   MIN(PROJ_MAXHOURS), MAX(PROJ_MAXHOURS), SUM(PROJ_MAXHOURS)
FROM     PROJECT
WHERE    PROJ_ID < 1500;
```
Output:

75.00	145.00	358.00

column names can't be mixed with built-in functions

NOT ALLOWED!!!!!!

~~SELECT PROJ_MAXHOURS, SUM(PROJ_MAXHOURS~~
~~FROM PROJECT~~
~~WHERE PROJ_ID < 1500;~~

SQL Built-In Functions and Grouping:

- Built-in functions can be applied to groups of rows of data.
  - **GROUP BY** phrase is used to tell the DBMS to sort the table by the named column and then to apply the built-in function to group of rows having the same value of the name column.
  - Built-in function & grouping column can appear in SELECT statement together (**this is the only time**)
  - Key word for GROUP BY: "**each**"

SQL Built-In Functions and Grouping:

Query Statement:

The <u>number of</u> employees in <u>each</u> department.

SQL Statement:

```
SELECT      EMP_DEPT, COUNT(*)
FROM        EMPLOYEE
GROUP BY    EMP_DEPT;
```

Output:

Accounting	2
Marketing	2
Finance	1
Info Systems	2

*****GROUP BY:* can't group by a column that is not stated in the SELECT statement.****

SQL Built-In Functions and Grouping:

- Restrictions to groups can be stated with the HAVING clause.

Query Statement:

The <u>number of</u> employees in <u>each</u> department with at least two members.

SQL Statement:

```
SELECT      EMP_DEPT, COUNT(*)
FROM        EMPLOYEE
GROUP BY    EMP_DEPT
HAVING      COUNT(*) > 1;
```

Output:

Accounting	2
Marketing	2
Info Systems	2

*****HAVING:* this states conditions applying to the group and is optional, if included must follow the GROUP BY phrase.****

Summary (PART ONE)

- Reading Specified Columns from single Table
- Reading Specified Rows from a single Table
- Reading Specified Columns from single Table and Specified Rows from a single Table
- Ranges, Wildcards, and Nulls in Where Clauses
- Sorting the Results
- SQL Built-In Functions