Data 301 Data Analytics Database Part II: SQL

Dr. Irene Vrbik

University of British Columbia Okanagan
irene.vrbik@ubc.ca

Intro

- So far we've learnt how to manipulate data stored in a relational database using:

  `CREATE` *to create new tables*
  `INSERT` *to insert new rows*
  `UPDATE` *to edit rows in a table*
  `ALTER` *to edit columns in a table*
  `DELETE` *for deleting rows*
  `DROP TABLE` *for deleting tables*

- Today we'll look at some examples for *querying* data from a database.

SQL Queries using SELECT

A query in SQL has the form:

```
SELECT <list of columns separated by commas>
FROM <list of tables>
WHERE <filter conditions>
GROUP BY <columns>
ORDER BY <columns> ASC/DESC
```

Notes:

1. Separate the list of columns/expressions and list of tables by commas.
2. The "*" is used to select all columns.
3. Only `SELECT` required. `FROM, WHERE, GROUP BY, ORDER BY` are optional.

Doing a query in SQL mode

- To see how to do a basic `SELECT` query in Microsoft Access, see 1:35-2:25 in this YouTube video.
- To see how to do a basic `SELECT` query in LibreOffice base, see this YouTube demo.

Example Data

Example Data

emp Table

eno	ename	bdate	title	salary	supereno	dno
E1	J. Doe	01-05-75	EE	30000	E2	null
E2	M. Smith	06-04-66	SA	50000	E5	D3
E3	A. Lee	07-05-66	ME	40000	E7	D2
E4	J. Miller	09-01-50	PR	20000	E6	D3
E5	B. Casey	12-25-71	SA	50000	E8	D3
E6	L. Chu	11-30-65	EE	30000	E7	D2
E7	R. Davis	09-08-77	ME	40000	E8	D1
E8	J. Jones	10-11-72	SA	50000	null	D1

workson Table

eno	pno	resp	hours
E1	P1	Manager	12
E2	P1	Analyst	24
E2	P2	Analyst	6
E3	P3	Consultant	10
E3	P4	Engineer	48
E4	P2	Programmer	18
E5	P2	Manager	24
E6	P4	Manager	48
E7	P3	Engineer	36

proj Table

pno	pname	budget	dno
P1	Instruments	150000	D1
P2	DB Develop	135000	D2
P3	Budget	250000	D3
P4	Maintenance	310000	D2
P5	CAD/CAM	500000	D2

dept Table

dno	dname	mgreno
D1	Management	E8
D2	Consulting	E7
D3	Accounting	E5
D4	Development	null

SQL: Retrieving Only Some Columns

The **projection operation** creates a new table that has some of the columns of the input table.

In SQL, provide the table in the `FROM` clause and the fields in the output in the `SELECT`.

Example: Return only the `eno` field from the `emp` table:

```
SELECT eno FROM emp
```

emp Table

eno	ename	bdate	title	salary	supereno	dno
E1	J. Doe	01-05-75	EE	30000	E2	null
E2	M. Smith	06-04-66	SA	50000	E5	D3
E3	A. Lee	07-05-66	ME	40000	E7	D2
E4	J. Miller	09-01-50	PR	20000	E6	D3
E5	B. Casey	12-25-71	SA	50000	E8	D3
E6	L. Chu	11-30-65	EE	30000	E7	D2
E7	R. Davis	09-08-77	ME	40000	E8	D1
E8	J. Jones	10-11-72	SA	50000	null	D1

Result

eno
E1
E2
E3
E4
E5
E6
E7
E8

SQL Projection Examples

emp Table

eno	ename	title	salary
E1	J. Doe	EE	30000
E2	M. Smith	SA	50000
E3	A. Lee	ME	40000
E4	J. Miller	PR	20000
E5	B. Casey	SA	50000
E6	L. Chu	EE	30000
E7	R. Davis	ME	40000
E8	J. Jones	SA	50000

```
SELECT eno,ename
FROM   emp
```

eno	ename
E1	J. Doe
E2	M. Smith
E3	A. Lee
E4	J. Miller
E5	B. Casey
E6	L. Chu
E7	R. Davis
E8	J. Jones

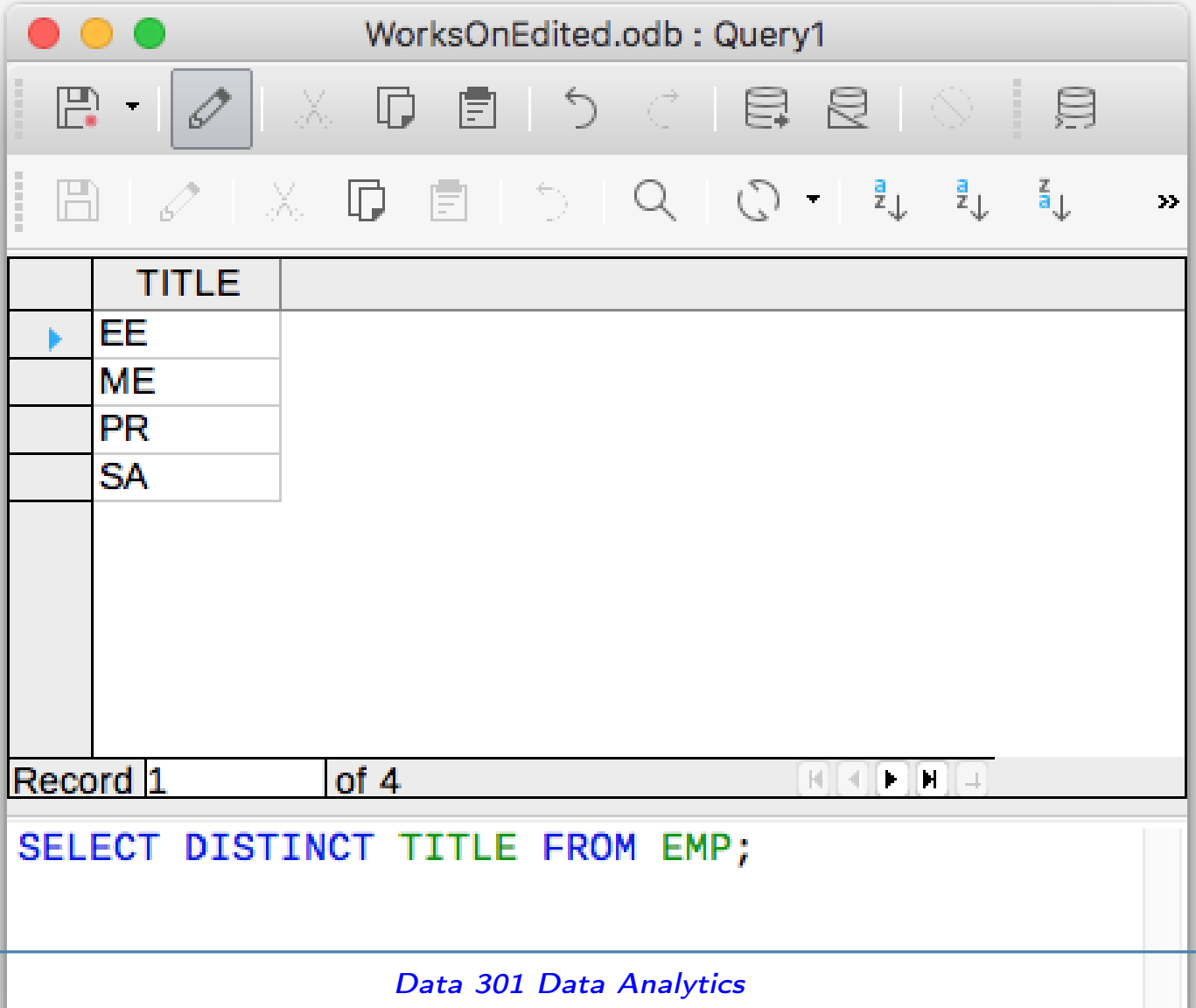
```
SELECT title
FROM   emp
```

title
EE
SA
ME
PR
SA
EE
ME
SA

Notice

1. Duplicates are not removed during SQL projection.
2. `SELECT *` will return all columns.

To return only the distinct values from the previous example, use DISTINCT:



Example 5.1

Given this table and the following query:

```
Select eno, ename, salary
From emp
```

How many columns are returned?

A) 0 **B)** 1 **C)** 2 **D)** 3 **E)** 4

emp Table

eno	ename	title	salary
E1	J. Doe	EE	30000
E2	M. Smith	SA	50000
E3	A. Lee	ME	40000
E4	J. Miller	PR	20000
E5	B. Casey	SA	50000
E6	L. Chu	EE	30000
E7	R. Davis	ME	40000
E8	J. Jones	SA	50000

Answer

Given this table and the following query:

```
Select eno, ename, salary
From emp
```

How many columns are returned?

A) 0 B) 1 C) 2 D) *3* E) 4

emp Table

eno	ename	title	salary
E1	J. Doe	EE	30000
E2	M. Smith	SA	50000
E3	A. Lee	ME	40000
E4	J. Miller	PR	20000
E5	B. Casey	SA	50000
E6	L. Chu	EE	30000
E7	R. Davis	ME	40000
E8	J. Jones	SA	50000

Example 5.2

Given this table and the following query:

```
Select salary
From emp
```

How many *rows* are returned?

A) 0 **B)** 1 **C)** 2 **D)** 4 **E)** 8

emp Table

eno	ename	title	salary
E1	J. Doe	EE	30000
E2	M. Smith	SA	50000
E3	A. Lee	ME	40000
E4	J. Miller	PR	20000
E5	B. Casey	SA	50000
E6	L. Chu	EE	30000
E7	R. Davis	ME	40000
E8	J. Jones	SA	50000

Answer

Given this table and the following query:

```
Select salary
From emp
```

How many *rows* are returned?

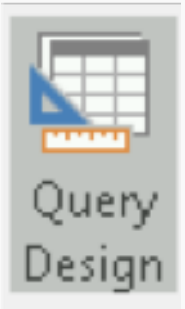
A) 0 B) 1 C) 2 D) 4 E) *8*

emp Table

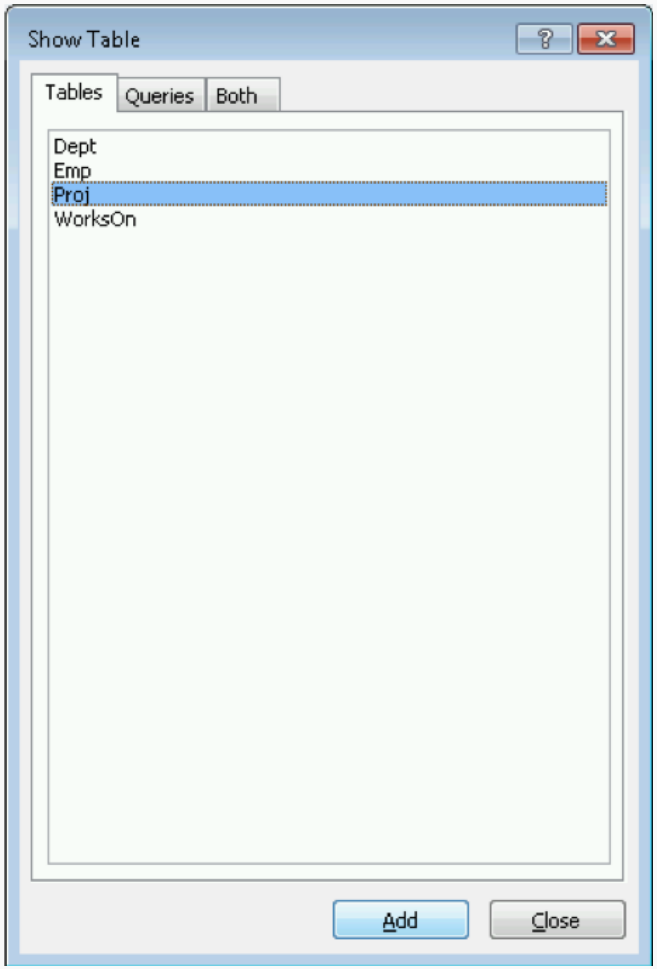
eno	ename	title	salary
E1	J. Doe	EE	30000
E2	M. Smith	SA	50000
E3	A. Lee	ME	40000
E4	J. Miller	PR	20000
E5	B. Casey	SA	50000
E6	L. Chu	EE	30000
E7	R. Davis	ME	40000
E8	J. Jones	SA	50000

Building a SELECT SQL Query in Microsoft Access

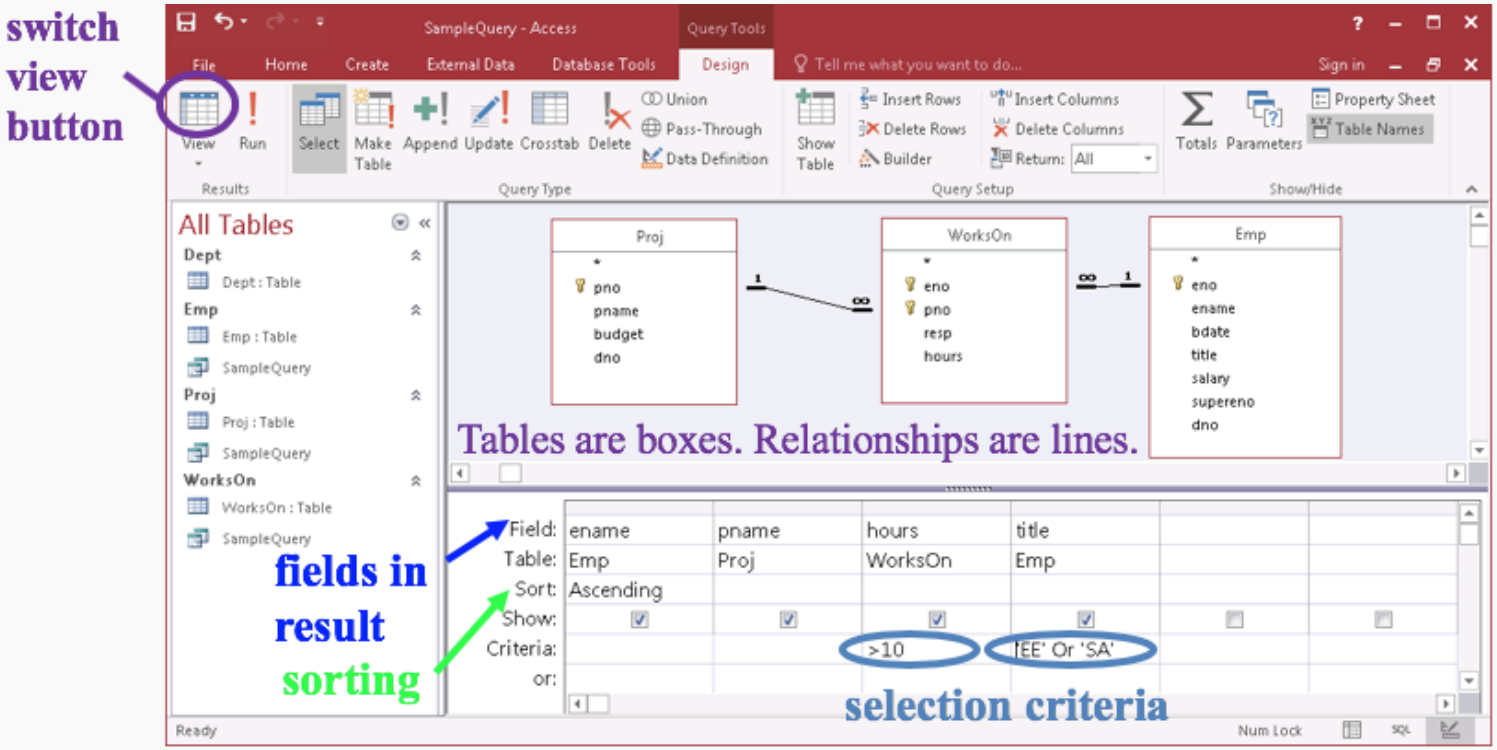
Under **Create Tab**, click on **Query Design**.



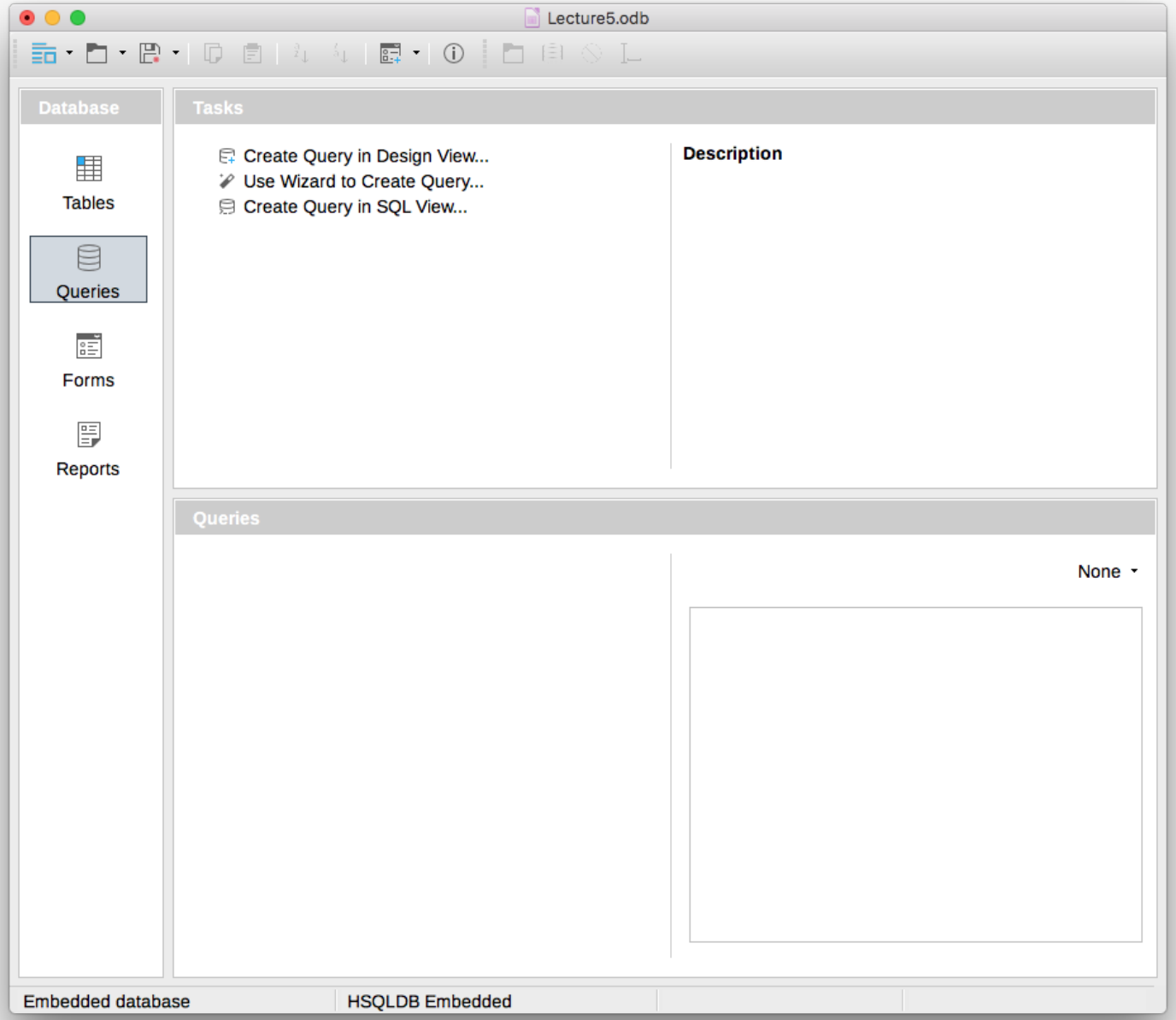
Access will pop-up a window asking what table(s) you wish to query. Select one or more.



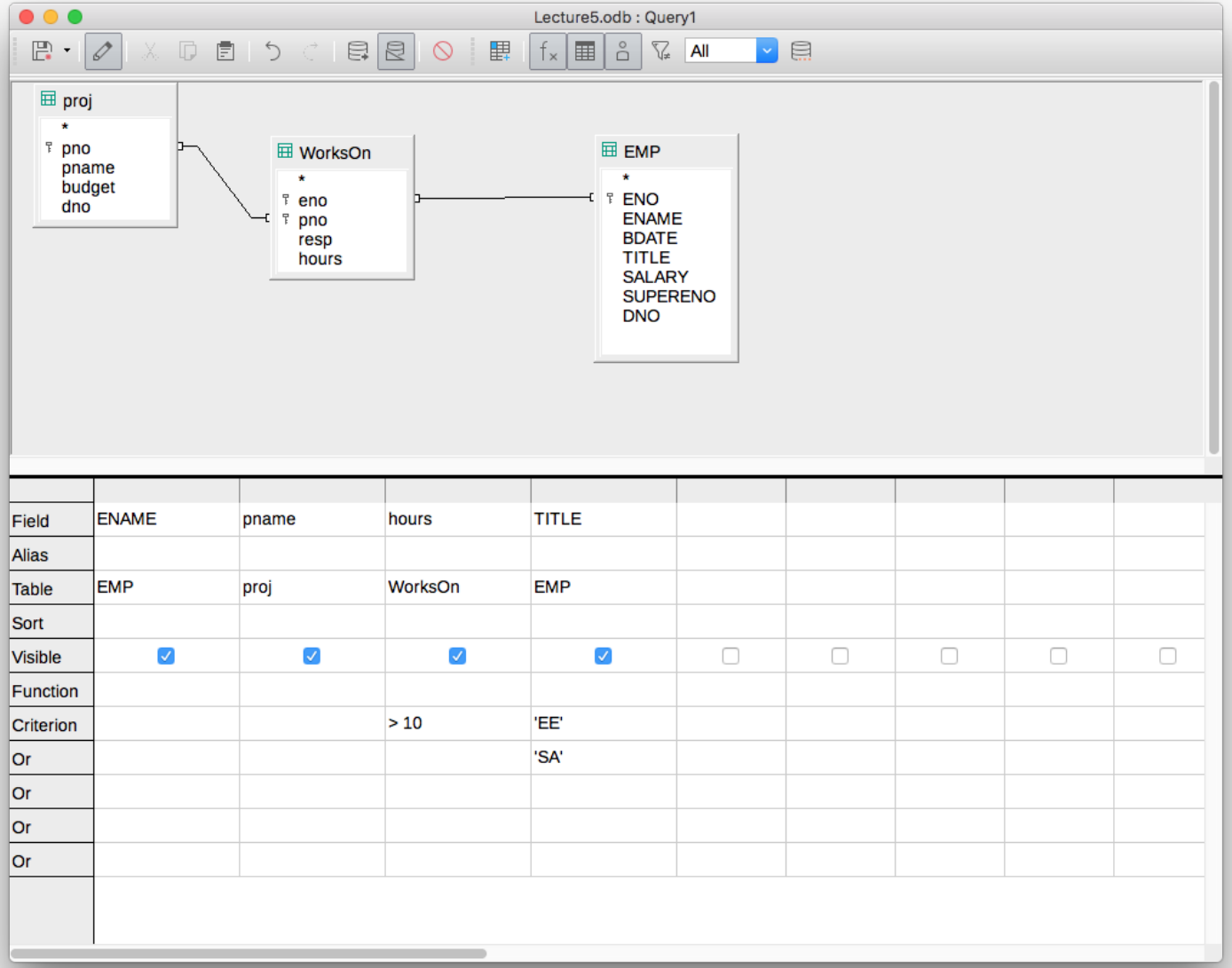
Microsoft Access Query Interface



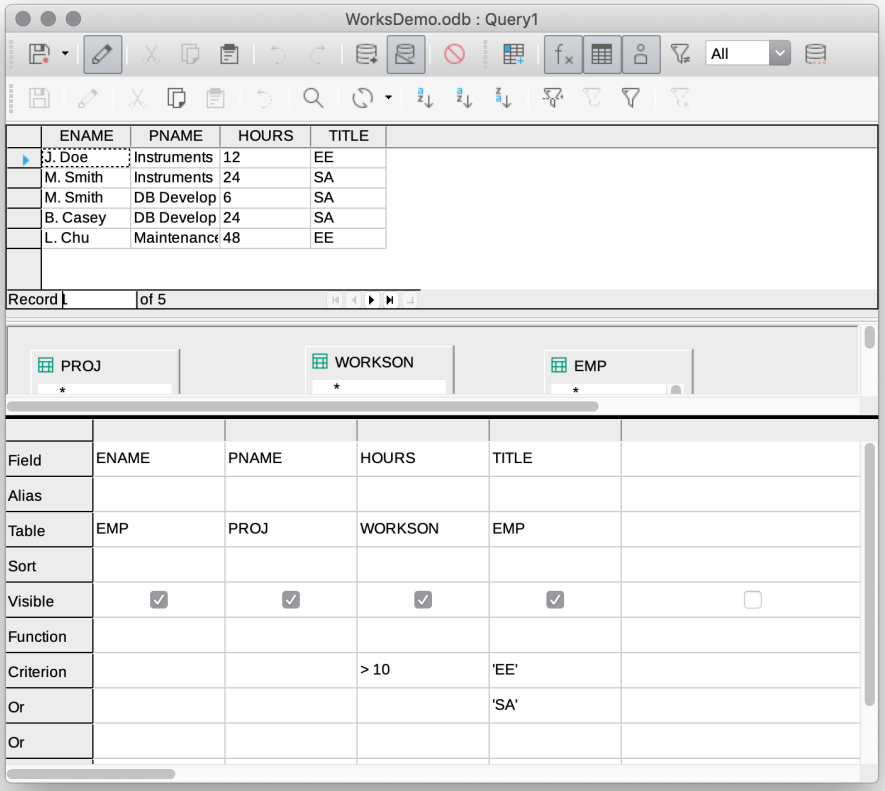
LibreOffice Query Interface



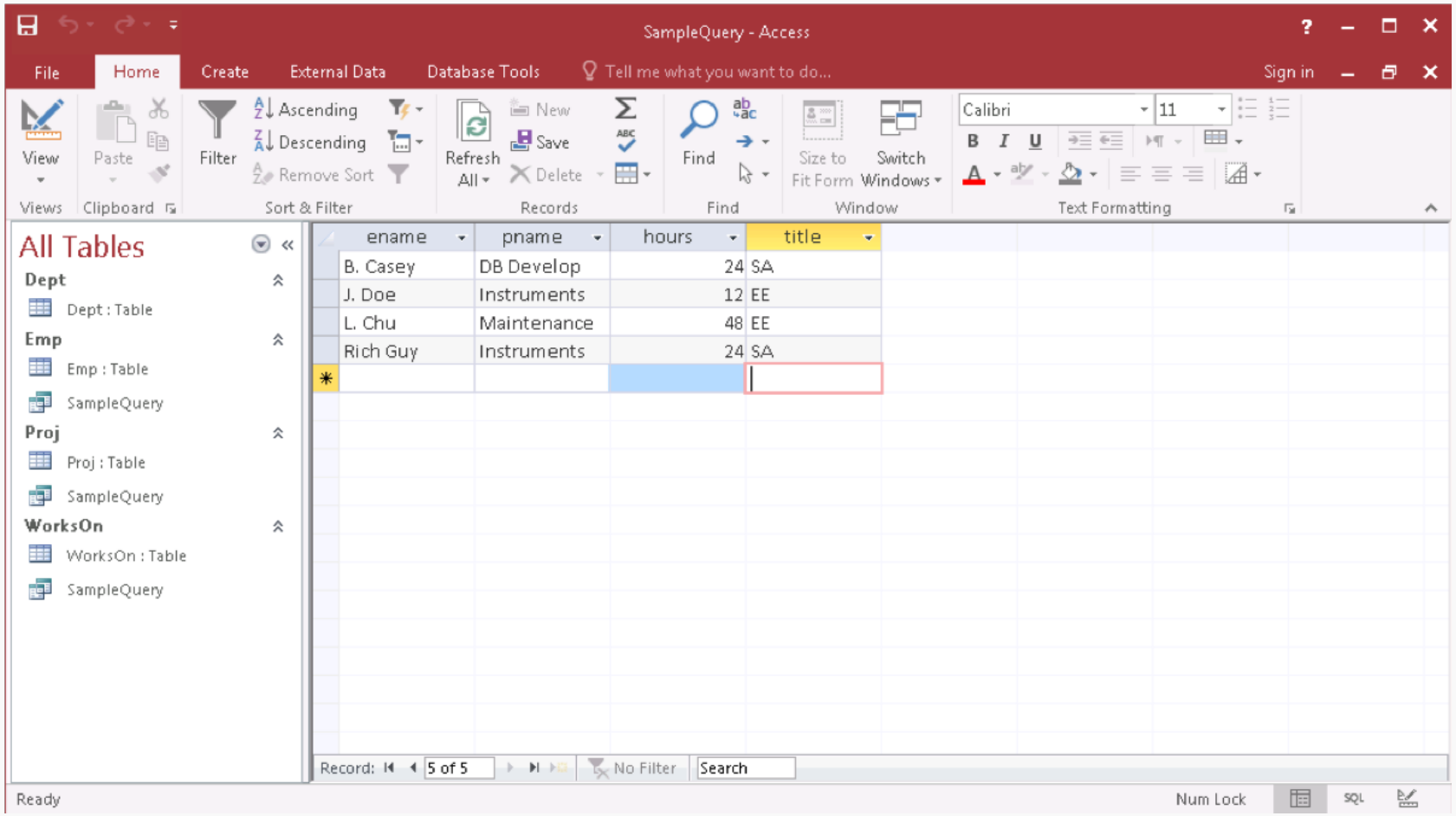
LibreOffice Query Interface



- If you hit the Run command (in LibreOffice base, Run in Access) and results will instantly appear in a data sheet view.
- We can save these results in Access by clicking the "Save" icon Toolbar. Give your query an identifying name, eg “Query1" for assignment question 1.

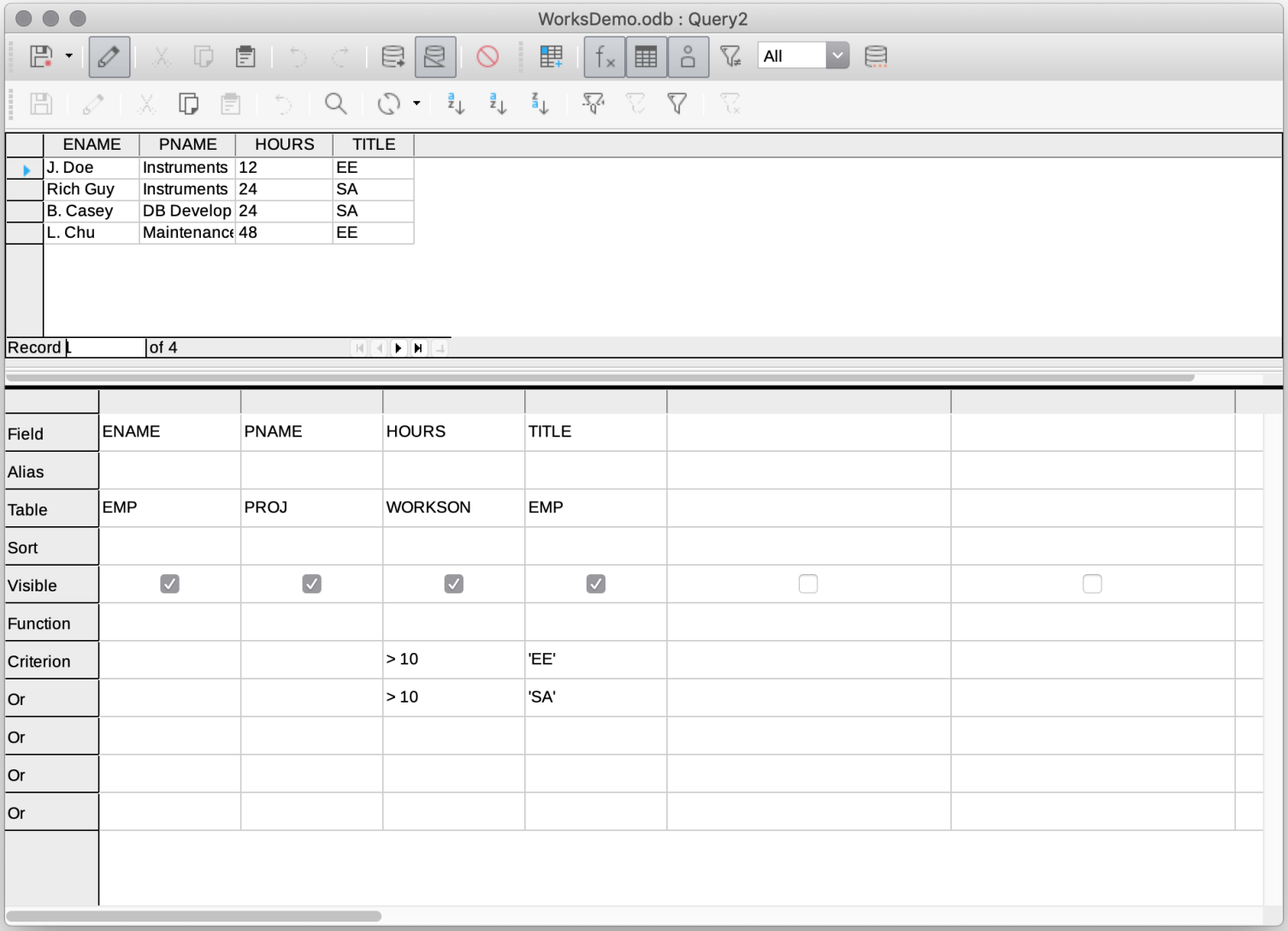


Microsoft Access Data Sheet View



LibreOffice Preview

To view the table: **View** →**Preview** or F5 or Run the Query



SQL Query View in LibreOffice

The design view window in Microsoft Access and LibreOffice provide an easy way of creating SQL queries

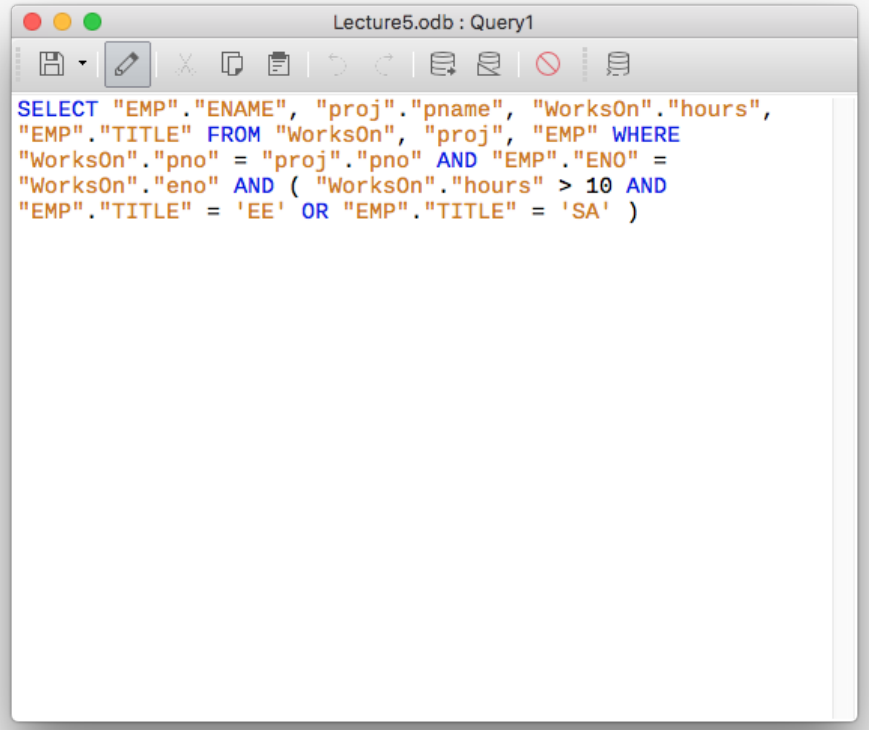
This may be referred to as QBE or Query By Example

Under the hood, these programs are creating SQL code and runs it to give us our result set.

To view that code in LibreOffice, we go to **View** →**Switch Design View On/Off**.

- 'Off' will show the query in SQL
- 'On' will show the query graphical

LibreOffice Query Views



Notice how the SQL code is not formatted very pretty. We can edit the white space, however, every time we close it, it will just get ugly again.

LibreOffice Query Views

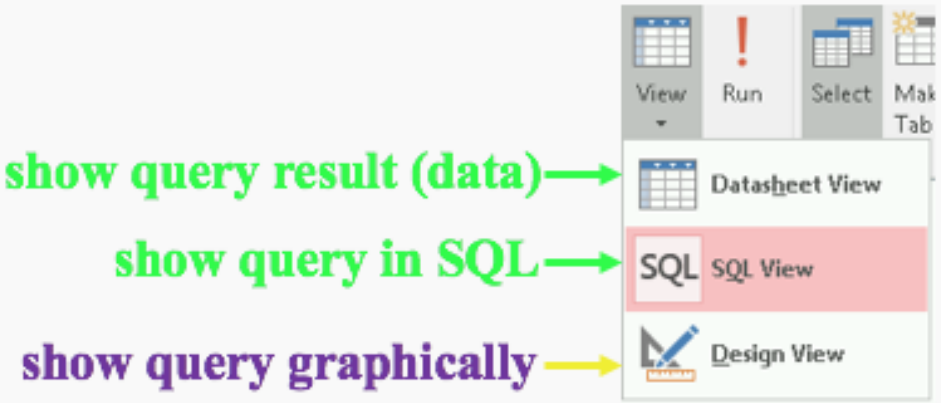
Lecture5.odb : Query1

```
SELECT
    "EMP"."ENAME",
    "proj"."pname",
    "WorksOn"."hours",
    "EMP"."TITLE"
FROM "WorksOn",
    "proj",
    "EMP"
WHERE "WorksOn"."pno" = "proj"."pno"
    AND "EMP"."ENO" = "WorksOn"."eno"
        AND ( "WorksOn"."hours" > 10
            AND "EMP"."TITLE" = 'EE'
                OR "EMP"."TITLE" = 'SA' )
```

Notice how the SQL code is not formatted very pretty. We can edit the white space, however, every time we close it, it will just get ugly again.

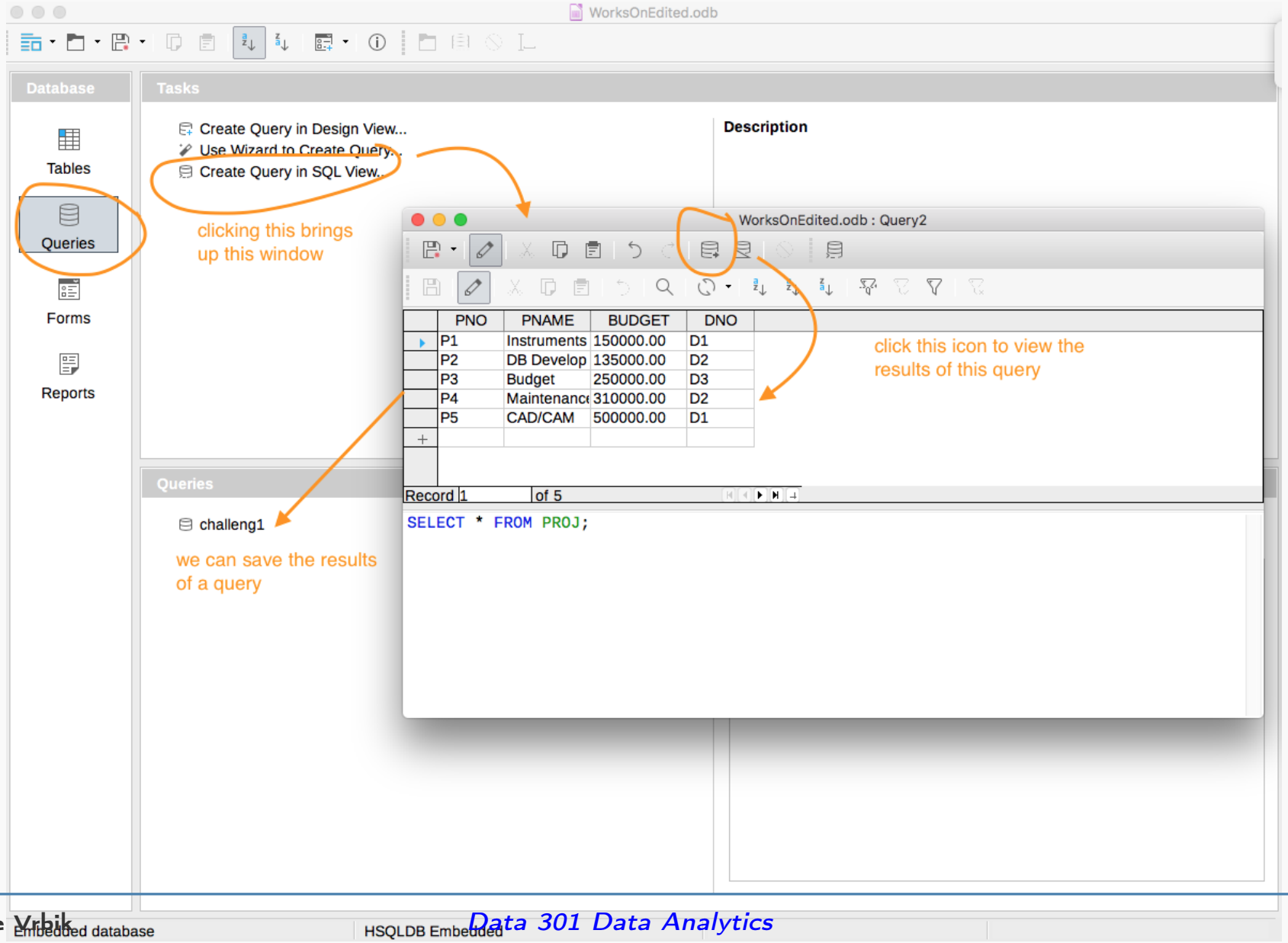
Microsoft Access Queries in SQL View

You may view your data, your query graphically, or your query in SQL.



For exam purposes, we will be needing to familiarize ourselves more with the SQL View.

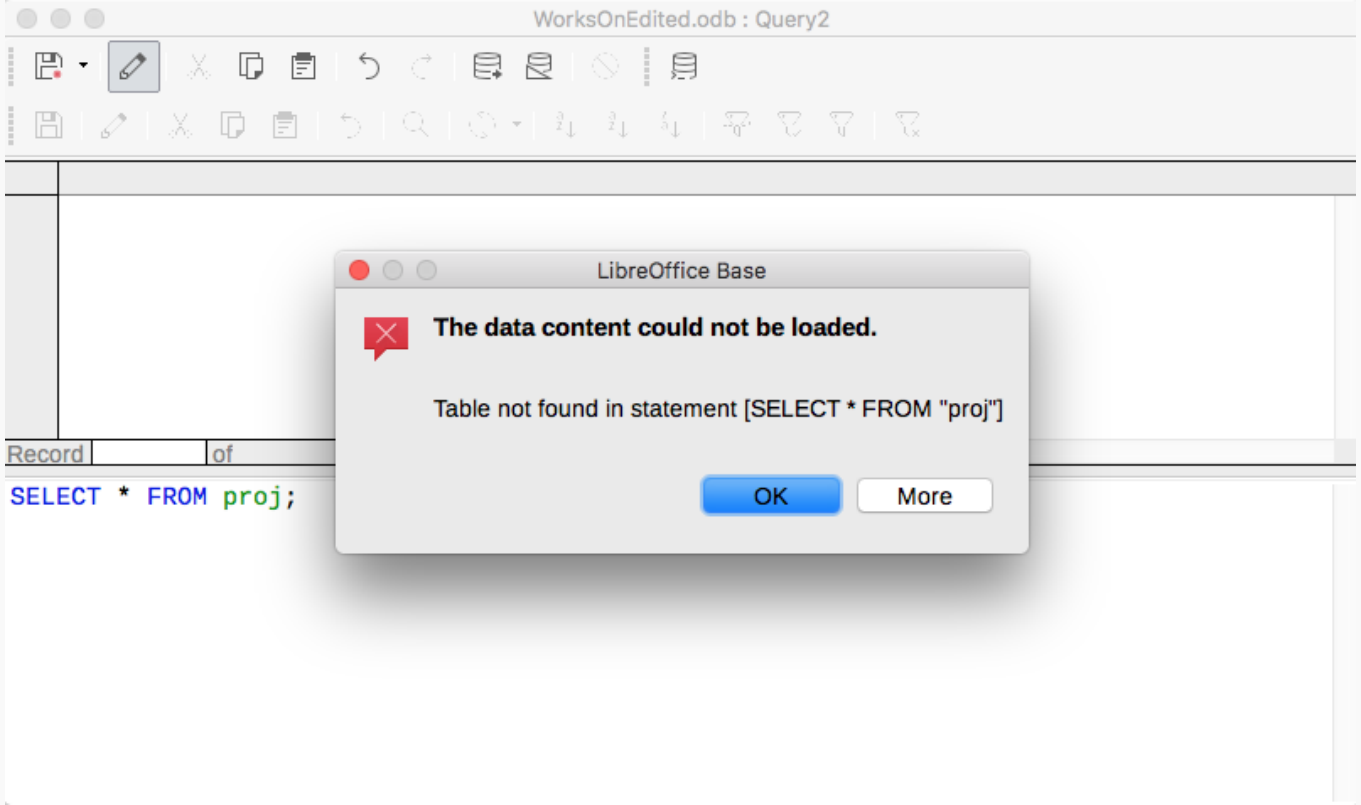
LibreOffice base Queries in SQL View



SQL Query View in LibreOffice

Warning

Unlike the SQL statements from last lecture, the SQL view mode for queries is case sensitive. That is, it will not convert all of our lower case to upper case text as it did before (annoying I know!)



Try it: SQL SELECT and Projection

Example 5.3

Using the `proj` table, write these three queries:

- Show all rows and all columns.
- Show all rows but only the `pno` column.
- Show all rows but only the `pno` and `budget` columns.

Retrieving only some of the rows

The **selection operation** creates a new table with some of the rows of the input table.

A *condition* specifies which rows are in the new table. This condition is similar to a filter in Excel.

Eg. the following algorithm scans each tuple and checks if it satisfies the condition in the WHERE clause.

```
SELECT * FROM proj WHERE dno = 'D2';
```

proj Table

pno	pname	budget	dno
P1	Instruments	150000	D1
P2	DB Develop	135000	D2
P3	Budget	250000	D3
P4	Maintenance	310000	D2
P5	CAD/CAM	500000	D2

Result

pno	pname	budget	dno
P2	DB Develop	135000	D2
P4	Maintenance	310000	D2
P5	CAD/CAM	500000	D2

Selection Conditions

- The condition in a selection statement specifies which rows are included.
- It has the general form of an `if` statement.
- The condition may consist of attributes, constants, comparison operators (`<, >, =, !=, <=, >=`), and logical operators (`AND, OR, NOT`).
- To check for `NULL`s, use `IS NULL`[1] which is different that checking if its equal to the empty string `=''`. General syntax:

```
SELECT column_names
FROM table_name
WHERE column_name IS NULL;
```

[1] `IS NOT NULL` is used to check for non null values

SQL Selection Examples

emp Table

eno	ename	title	salary
E1	J. Doe	EE	30000
E2	M. Smith	SA	50000
E3	A. Lee	ME	40000
E4	J. Miller	PR	20000
E5	B. Casey	SA	50000
E6	L. Chu	EE	30000
E7	R. Davis	ME	40000
E8	J. Jones	SA	50000

```
SELECT *
FROM emp
WHERE title = 'EE'
```

eno	ename	title	salary
E1	J. Doe	EE	30000
E6	L. Chu	EE	30000

```
SELECT *
FROM emp
WHERE salary > 35000
OR title = 'PR'
```

eno	ename	title	salary
E2	M. Smith	SA	50000
E3	A. Lee	ME	40000
E4	J. Miller	PR	20000
E5	B. Casey	SA	50000
E7	R. Davis	ME	40000
E8	J. Jones	SA	50000

Selection Question

Example 5.4

Given the emp table and the following query, how many rows are returned?

```
SELECT *
FROM emp
WHERE title = 'SA'
```

A) 0 B) 1 C) 2 D) 3 E) >3

emp Table

eno	ename	title	salary
E1	J. Doe	EE	30000
E2	M. Smith	SA	50000
E3	A. Lee	ME	40000
E4	J. Miller	PR	20000
E5	B. Casey	SA	50000
E6	L. Chu	EE	30000
E7	R. Davis	ME	40000
E8	J. Jones	SA	50000

Selection Question

Answer

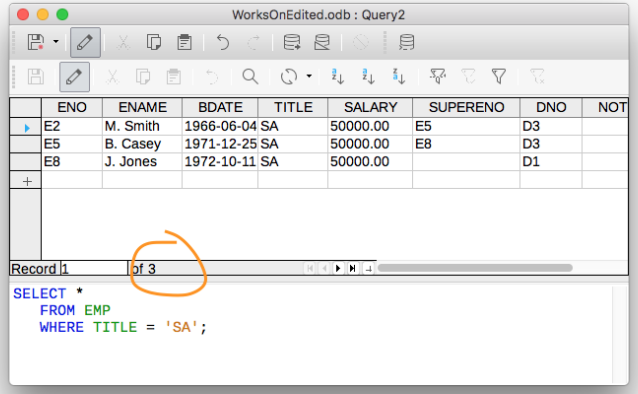
Given the emp table and the following query, how many rows are returned?

```
SELECT *
FROM emp
WHERE title = 'SA'
```

A) 0 B) 1 C) 2 D) 3 E) >3

emp Table

eno	ename	title	salary
E1	J. Doe	EE	30000
E2	M. Smith	SA	50000
E3	A. Lee	ME	40000
E4	J. Miller	PR	20000
E5	B. Casey	SA	50000
E6	L. Chu	EE	30000
E7	R. Davis	ME	40000
E8	J. Jones	SA	50000



Selection Question2

Example 5.5

Given the emp table and the following query, how many rows are returned?

```
SELECT *
FROM emp
WHERE salary > 40000
  OR title='PR'
```

A) 0

B) 1

C) 3

D) 4

E) > 4

emp Table

eno	ename	title	salary
E1	J. Doe	EE	30000
E2	M. Smith	SA	50000
E3	A. Lee	ME	40000
E4	J. Miller	PR	20000
E5	B. Casey	SA	50000
E6	L. Chu	EE	30000
E7	R. Davis	ME	40000
E8	J. Jones	SA	50000

Selection Question 2

Answer

Given the emp table and the following query, how many rows are returned?

```
SELECT *
FROM emp
WHERE salary > 40000
  OR title='PR'
```

A) 0

B) 1

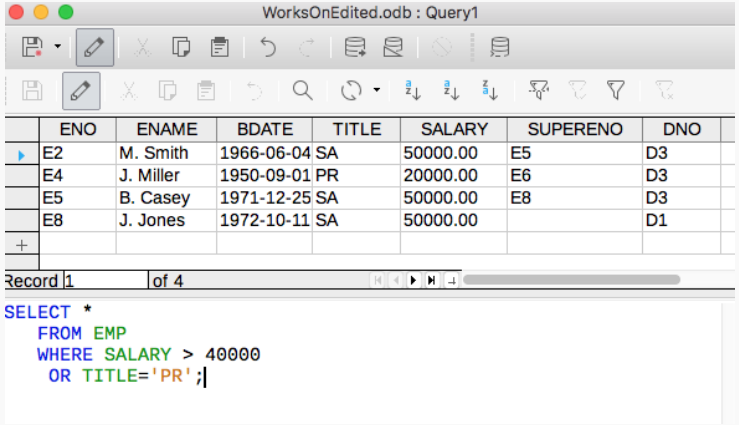
C) 3

D) *4*

E) > 4

emp Table

eno	ename	title	salary
E1	J. Doe	EE	30000
E2	M. Smith	SA	50000
E3	A. Lee	ME	40000
E4	J. Miller	PR	20000
E5	B. Casey	SA	50000
E6	L. Chu	EE	30000
E7	R. Davis	ME	40000
E8	J. Jones	SA	50000



Selection Question 3

emp Table

eno	ename	title	salary
E1	J. Doe	EE	30000
E2	M. Smith	SA	50000
E3	A. Lee	ME	40000
E4	J. Miller	PR	20000
E5	B. Casey	SA	50000
E6	L. Chu	EE	30000
E7	R. Davis	ME	40000
E8	J. Jones	SA	50000

Example 5.6

Given the emp table and the following query, how many rows are returned?

```
SELECT *
FROM emp
WHERE salary >= 40000
  AND ename > 'C'
```

A) 0

B) 1

C) 2

D) 3

E) ≥ 4

Selection Question 3

Answer

Given the emp table and the following query, how many rows are returned?

```
SELECT *
FROM emp
WHERE salary >= 40000
  AND ename > 'C'
```

A) 0

B) 1

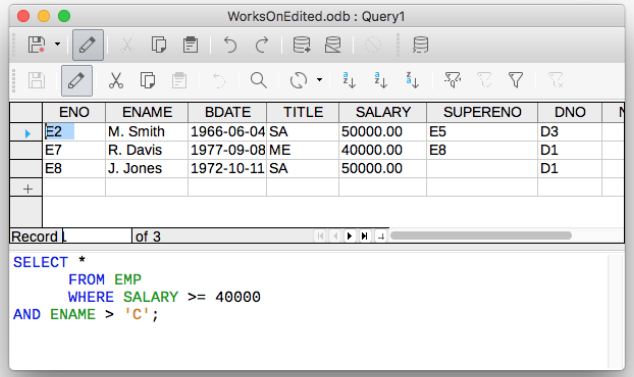
C) 2

D) *3*

E) ≥ 4

emp Table

eno	ename	title	salary
E1	J. Doe	EE	30000
E2	M. Smith	SA	50000
E3	A. Lee	ME	40000
E4	J. Miller	PR	20000
E5	B. Casey	SA	50000
E6	L. Chu	EE	30000
E7	R. Davis	ME	40000
E8	J. Jones	SA	50000



Try It: SQL SELECT and Filtering Rows

Example 5.7

Using the `proj` table, write these three queries:

- Return all projects with `budget > $250000`.
- Show the `pno` and `pname` for projects in `dno = 'D1'`.
- Show `pno` and `dno` for projects in `dno='D1'` or `dno='D2'`.

Join Example for Combining Tables

A **join** combines two tables by matching columns in each table.

eno	pno	resp	dur
E1	P1	Manager	12
E2	P1	Analyst	24
E2	P2	Analyst	6
E3	P4	Engineer	48
E5	P2	Manager	24
E6	P4	Manager	48
E7	P3	Engineer	36
E7	P4	Engineer	23

Figure: `workson`

```
SELECT *
FROM WorksOn
INNER JOIN Proj
  ON WorksOn.pno = Proj.pno
```

eno	pno	resp	dur	P.pno	pname	budget
E1	P1	Manager	12	P1	Instruments	150000
E2	P1	Analyst	24	P1	Instruments	150000
E2	P2	Analyst	6	P2	DB Develop	135000
E3	P4	Engineer	48	P4	Maintenance	310000
E5	P2	Manager	24	P2	DB Develop	135000
E6	P4	Manager	48	P4	Maintenance	310000
E7	P3	Engineer	36	P3	CAD/CAM	250000
E7	P4	Engineer	23	P4	Maintenance	310000

pno	pname	budget
P1	Instruments	150000
P2	DB Develop	135000
P3	CAD/CAM	250000
P4	Maintenance	310000
P5	CAD/CAM	500000

Figure: `proj`

The general syntax is:

```
SELECT <columns>
FROM
R
<type of join> JOIN
S
ON R.<colname>= S.<colname>;
```

Since joining tables often result in repeated field (ie. columns) we distinguish between them using `table_name.column_name`.

N.B. the related columns will not necessarily have the same name (but oftentimes they will!)

Joining Tables

When connecting tables R and S, there are four types of joins:

(INNER) JOIN row in result for each row of R that matches a row of S

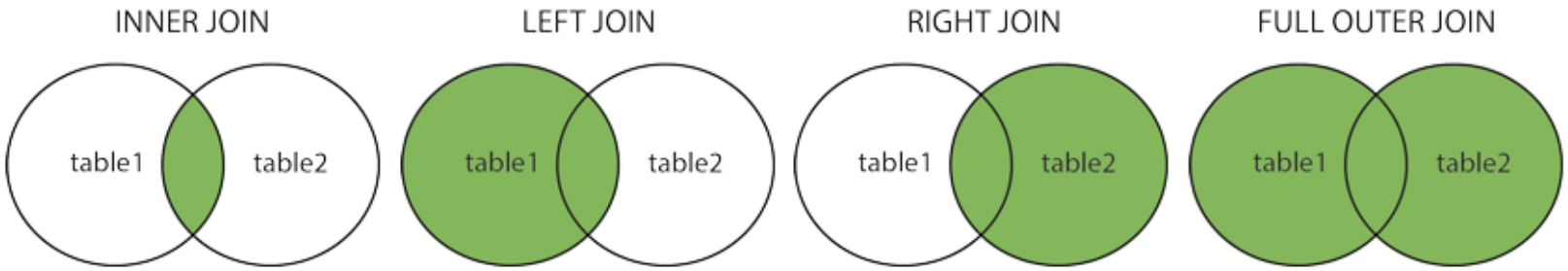
LEFT (OUTER) JOIN row in result for each row of R that matches a row of S OR a row of R that does not match anything in S

RIGHT (OUTER) JOIN row in result for each row of R that matches a row of S OR a row of S that does not match anything in R

FULL OUTER JOIN row in result for each row of R that matches a row of S OR a row of R that does not match anything in S OR a row of S that does not match anything in R

w3schools is another good resource for learning SQL. Here is a helpful representation of the different types of joins (where R = table1 and S=right table2).

- **(INNER) JOIN**: Returns records that have matching values in both tables
- **LEFT (OUTER) JOIN**: Returns all records from the left table, and the matched records from the right table
- **RIGHT (OUTER) JOIN**: Returns all records from the right table, and the matched records from the left table
- **FULL (OUTER) JOIN**: Returns all records when there is a match in either left or right table



Join Example

```
SELECT * FROM Boys <>  JOIN Girls ON Boys.Bid = Girls.Gid;
```

Boys

Bid	BoyName
1	Joe
2	Steve
3	Fred
5	James

Boys INNER JOIN Girls

Bid	BoyName	Gid	GirlName
2	Steve	2	Jane
5	James	5	Fran

Boys LEFT OUTER JOIN Girls

Bid	BoyName	Gid	GirlName
1	Joe		
2	Steve	2	Jane
3	Fred		
5	James	5	Fran

Girls

Gid	GirlName
2	Jane
4	Sarah
5	Fran
6	Julie

Boys FULL OUTER JOIN Girls

Bid	BoyName	Gid	GirlName
1	Joe		
2	Steve	2	Jane
3	Fred		
		4	Sarah
5	James	5	Fran
		6	Julie

Boys RIGHT OUTER JOIN Girls

Bid	BoyName	Gid	GirlName
2	Steve	2	Jane
		4	Sarah
5	James	5	Fran
		6	Julie

Join Query with Selection Example

You can use join, selection, and projection in the same query. Recall:

- **projection** returns *columns* listed in `SELECT`
- **selection** filters out *rows* using condition in `WHERE`, and
- **join** combines *tables* in `FROM` using a condition.

Example 5.8

Return the employee names who are assigned to the 'Management' department.

emp Table

eno	ename	bdate	title	salary	supereno	dno
E1	J. Doe	01-05-75	EE	30000	E2	null
E2	M. Smith	06-04-66	SA	50000	E5	D3
E3	A. Lee	07-05-66	ME	40000	E7	D2
E4	J. Miller	09-01-50	PR	20000	E6	D3
E5	B. Casey	12-25-71	SA	50000	E8	D3
E6	L. Chu	11-30-65	EE	30000	E7	D2
E7	R. Davis	09-08-77	ME	40000	E8	D1
E8	J. Jones	10-11-72	SA	50000	null	D1

dept Table

dno	dname	mgreno
D1	Management	E8
D2	Consulting	E7
D3	Accounting	E5
D4	Development	null

Projection: only name field in result

```
SELECT ename
FROM   emp INNER JOIN dept
         ON emp.dno = dept.dno
WHERE  dname = 'Management';
```

tables in query joined together

Selection: filter rows

Result

ename
R. Davis
J. Jones

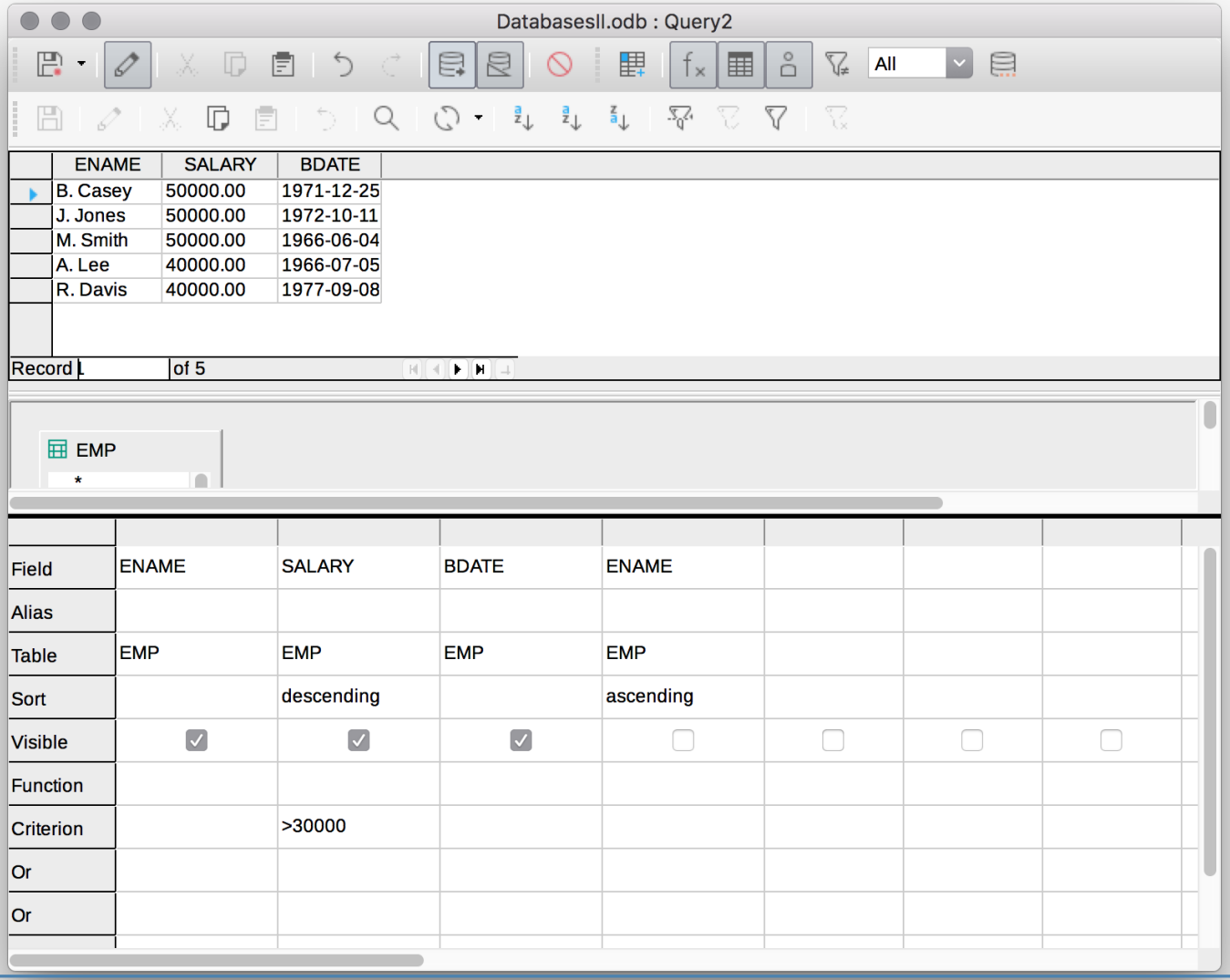
Ordering Result Data

The query result returned is not ordered on any column by default. We can order the data using the ORDER BY clause:

```
SELECT ename, salary, bdate
FROM emp
WHERE salary > 30000
  ORDER BY salary DESC, ename ASC;
```

- ASC sorts the data in ascending order, and DESC sorts it in descending order. The default is ASC.
- The order of sorted attributes matters; namely, the first column specified is sorted on first, then the second column is used to break any ties, etc.

In LibreOffice the order is done Left to right. To ensure that `salary` gets ordered first, add another hidden (Deselect **Visible**) column with `ename` to the right.



LIMIT and OFFSET

If you only want the first *N* rows, use a LIMIT clause:

```
SELECT ename, salary FROM emp
ORDER BY salary DESC LIMIT 5;
```

To start from a row besides the first, use OFFSET:

```
SELECT ename, salary FROM emp
ORDER BY salary DESC LIMIT 5 OFFSET 2;
```

- LIMIT improves performance by reducing amount of data processed and sent by the database system.
- OFFSET 0 is first row, so OFFSET 2 would return the 3rd row.

- LIMIT/OFFSET syntax is supported differently by systems.
- For example, Access uses

```
SELECT TOP 5 eno, salary FROM emp
```

Program	syntax
MySQL, PostgreSQL	LIMIT syntax
Oracle	ROWNUM field that can be filtered in WHERE
SQL Server	SELECT TOP N

(click the pink text above and here for more details.)

Try It: SQL SELECT with Joins and Ordering

Example 5.9

Write these three queries:

- Return all projects with `budget < $500000` sorted by `budget` descending.
- List only the top 5 employees by `salary` descending. Show only their `name` and `salary`.
- List each project `pno, dno, pname`, and `dname` ordered by `dno` ascending then `pno` ascending. Only show projects if department name > `'D'`. Note: This query will require a `join`.

SELECT Statement Execution Order

Order written:

```
SELECT < feilds >
FROM < left_table >
JOIN < right_table >
ON < join_condition >
WHERE < where_condition >
GROUPBY < group_by_list >
ORDERBY < order_by_list >
```

Order executed:

1. FROM clause
2. JOIN clause
3. WHERE clause
4. GROUP BY clause
5. SELECT clause
6. DISTINCT clause
7. ORDER BY clause

Read more about it here

Aggregate Queries and Functions

Several queries cannot be answered using the simple form of the `SELECT` statement. These queries require a summary calculation to be performed. For example:

- What is the maximum employee salary?
- What is the total number of hours worked on a project?
- How many employees are there in department 'D1'?

To answer these queries requires the use of **aggregate** functions. These functions operate on a single column of a table and return a single value.

Aggregate Functions

Five common aggregate functions are:

```
COUNT returns the number of values in a column
SUM returns the sum of the values in a column
AVG returns the average of the values in a column
MIN returns the smallest value in a column
MAX returns the largest value in a column
```

Notes:

1. `COUNT`, `MAX`, and `MIN` apply to all types of fields, whereas `SUM` and `AVG` apply to only numeric fields.
2. Except for `COUNT(*)` all functions ignore nulls. `COUNT(*)` returns the number of rows in the table.
3. Use `DISTINCT` to eliminate duplicates.

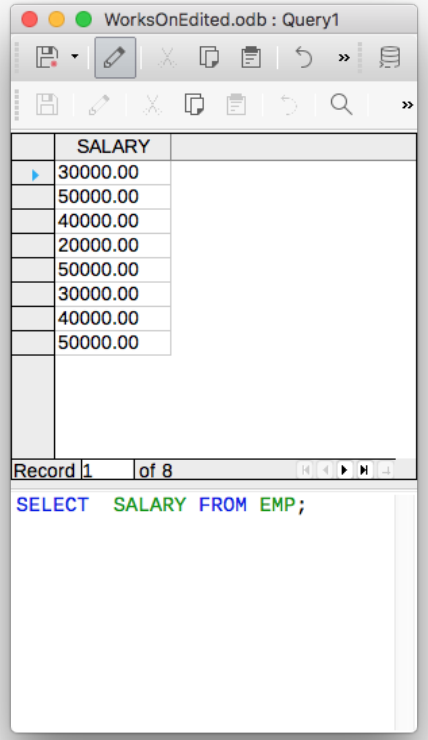
DISTINCT syntax

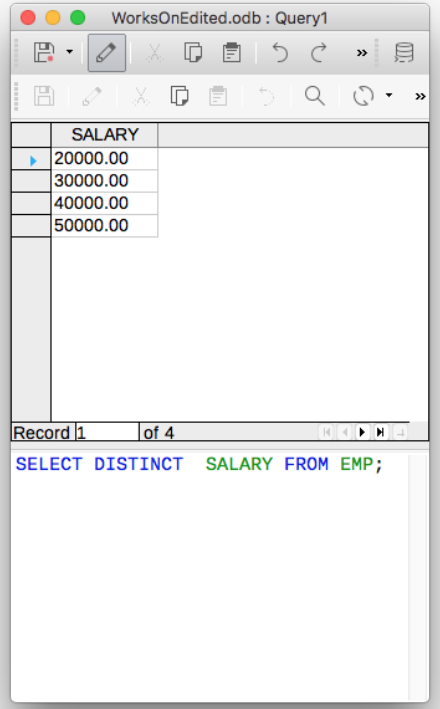
The following selects the *distinct* salaries from emp:

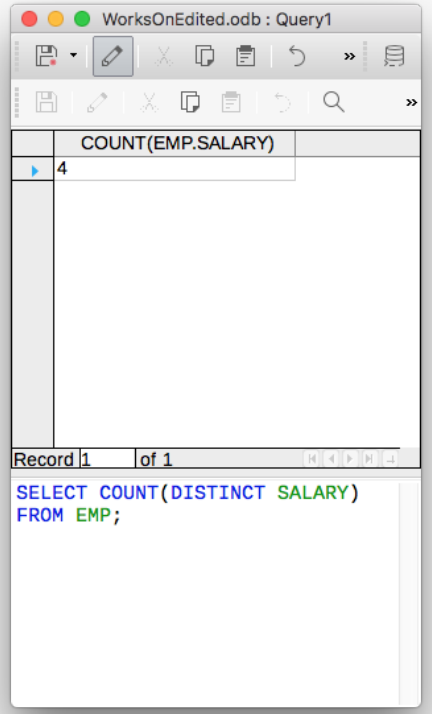
- `SELECT DISTINCT salary FROM Emp`

The following counts the number of distinct salaries from emp:

- `SELECT COUNT(DISTINCT salary) FROM Emp`

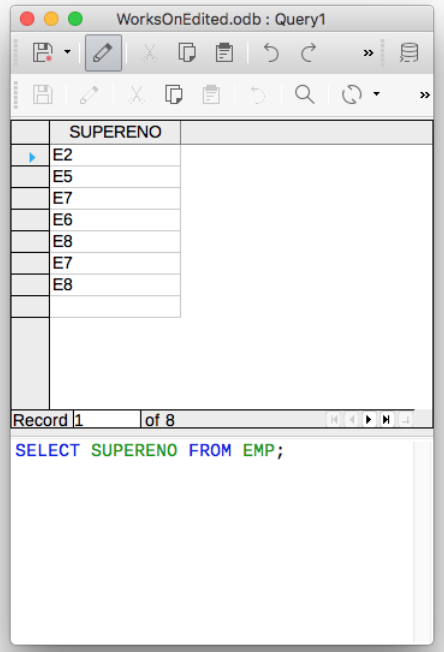


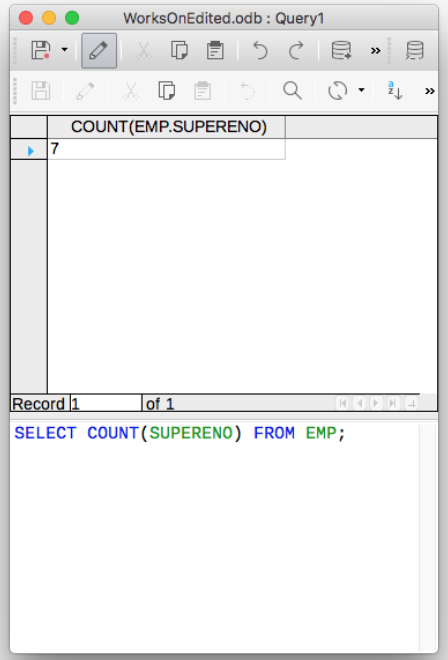


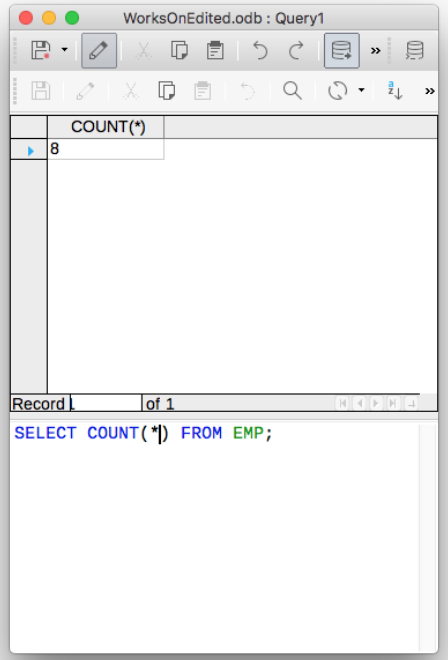


COUNT

Use `SELECT COUNT(*) FROM EMP;` to count all the rows including NULLS and duplicates.







Aggregate Function Example

Return the number of employees and their average salary.

```
SELECT COUNT(eno) AS numEmp, AVG(salary) AS avgSalary
FROM emp
```

numEmp	avgSalary
8	38750

Note: AS is used to rename a column in the output.
N.B. Aggregate functions are separated by commas just like any other field name.

GROUP BY Clause

Aggregate functions are most useful when combined with the GROUP BY clause.

The GROUP BY clause groups rows based on the values of the columns specified.

When used in combination with aggregate functions, the result is a table where each row consists of unique values for the GROUP BY attributes and the result of the aggregate functions applied to the rows of that group.

GROUP BY example

For each employee title, return the number of employees with that title.

```
SELECT title, COUNT(eno) AS numEmp
FROM emp
GROUP BY title
```

TITLE	numEmp
EE	2
SA	3
ME	2
PR	1

GROUP BY example

If a selected field is not aggregated by a function it has to be explicitily added to the GROUP BY clause!!

```
SELECT <A>, >B>, aggregateFun(X) AS varName
GROUP BY <A>, >B>
```

For example.

```
SELECT title, COUNT(eno) AS numEmp
FROM emp
GROUP BY title <- without this line we get an error
```

TITLE	numEmp
EE	2
SA	3
ME	2
PR	1

GROUP BY example

For each employee title, return the number of employees with that title, and the minimum, maximum, and average salary.

```
SELECT title, COUNT(eno) AS numEmp,
  MIN(salary) AS minSal,
  MAX(salary) AS maxSal, AVG(salary) AS avgSal
FROM emp
GROUP BY title
```

title	numEmp	minSal	maxSal	avgSal
EE	2	30000	30000	30000
SA	3	50000	50000	50000
ME	2	40000	40000	40000
PR	1	20000	20000	20000

GROUP BY Facts

1. You can group by multiple attributes. To be in the same group, all attribute values must be the same.
2. Any WHERE conditions are applied before the GROUP BY and aggregate functions are calculated.
3. A column name cannot appear in the SELECT part of the query unless it is part of an aggregate function or in the list of group by attributes.
4. There is a HAVING clause that is applied after the GROUP BY clause and aggregate functions are calculated to filter out groups.

SELECT Statement Execution Order

Order written:

SELECT < feilds >
FROM < left_table >
JOIN < right_table >
ON < join_condition >
WHERE < where_condition >
GROUPBY < group_by_list >
HAVING < having_condition >
ORDERBY < order_by_list >

Order executed:

1. FROM clause
2. JOIN clause
3. WHERE clause
4. GROUP BY clause
5. HAVING clause
6. SELECT clause
7. DISTINCT clause
8. ORDER BY clause

Read more about it here

WHERE filters *before* GROUP BY whereas HAVING filters after.

WorksOnEdited.odb : Query1

TITLE	numEmp	minSal	maxSal	avgSal
EE	2	30000	30000	30000
SA	3	50000	50000	50000
ME	2	40000	40000	40000
PR	1	20000	20000	20000

Record 1 of 4

```
SELECT  TITLE,
    COUNT(ENO) AS numEmp,
    MIN(SALARY) as minSal,
    MAX(SALARY) as maxSal,
    AVG(SALARY) AS avgSal
FROM     EMP
GROUP BY TITLE
;
```

WorksOnEdited.odb : Query1

TITLE	numEmp	minSal	maxSal	avgSal
EE	2	30000	30000	30000
SA	2	50000	50000	50000
PR	1	20000	20000	20000
ME	1	40000	40000	40000

Record 1 of 4

```
SELECT  TITLE,
    COUNT(ENO) AS numEmp,
    MIN(SALARY) as minSal,
    MAX(SALARY) as maxSal,
    AVG(SALARY) AS avgSal
FROM     EMP
WHERE ENAME > 'C'
GROUP BY TITLE
;
```

WorksOnEdited.odb : Query1

TITLE	numEmp	minSal	maxSal	avgSal
SA	3	50000	50000	50000
ME	2	40000	40000	40000

Record 1 of 2

```
SELECT  TITLE,
    COUNT(ENO) AS numEmp,
    MIN(SALARY) as minSal,
    MAX(SALARY) as maxSal,
    AVG(SALARY) AS avgSal
FROM     EMP
GROUP BY TITLE
HAVING MIN(SALARY) > 30000
;
```

GROUP BY Question

Example 5.10

Given the emp table and the following query, how many rows are returned?

```
SELECT title, SUM(salary)
FROM emp
GROUP BY title
```

A) 1 **B)** 2 **C)** 4 **D)** 8

emp Table

eno	ename	title	salary
E1	J. Doe	EE	30000
E2	M. Smith	SA	50000
E3	A. Lee	ME	40000
E4	J. Miller	PR	20000
E5	B. Casey	SA	50000
E6	L. Chu	EE	30000
E7	R. Davis	ME	40000
E8	J. Jones	SA	50000

GROUP BY Question

Answer

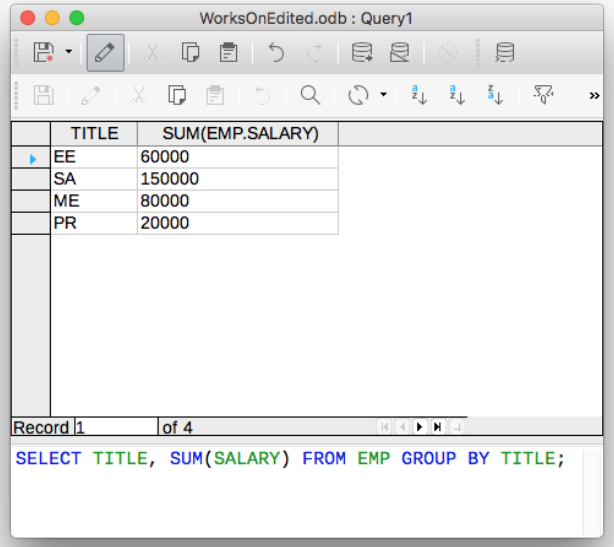
Given the emp table and the following query, how many rows are returned?

```
SELECT title, SUM(salary)
FROM emp
GROUP BY title
```

A) 1 B) 2 C) 4 D) 8

emp Table

eno	ename	title	salary
E1	J. Doe	EE	30000
E2	M. Smith	SA	50000
E3	A. Lee	ME	40000
E4	J. Miller	PR	20000
E5	B. Casey	SA	50000
E6	L. Chu	EE	30000
E7	R. Davis	ME	40000
E8	J. Jones	SA	50000



GROUP BY Question 2

Example 5.11

Given the workson table and the following query, how many rows are returned?

```
SELECT resp, pno, SUM(hours)
FROM workson
WHERE hours > 10
GROUP BY resp, pno
```

A) 9 B) 7 C) 5 D) 1 E) 0

workson Table

eno	pno	resp	hours
E1	P1	Manager	12
E2	P1	Analyst	24
E2	P2	Analyst	6
E3	P3	Consultant	10
E3	P4	Engineer	48
E4	P2	Programmer	18
E5	P2	Manager	24
E6	P4	Manager	48
E7	P3	Engineer	36

Figure: workson

GROUP BY Question

workson Table

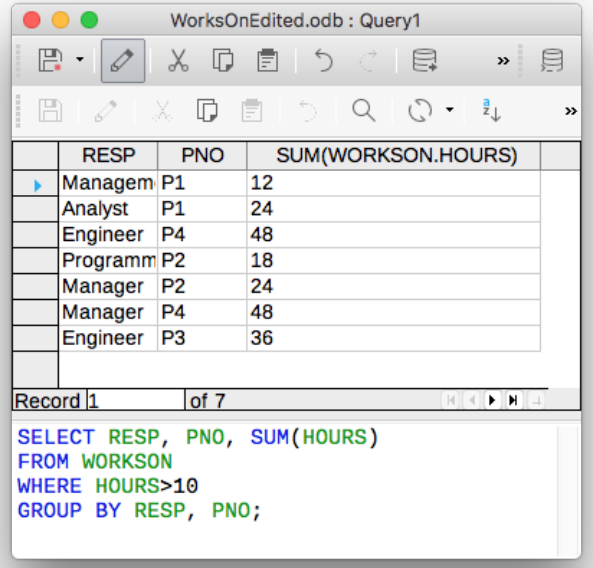
eno	pno	resp	hours
E1	P1	Manager	12
E2	P1	Analyst	24
E2	P2	Analyst	6
E3	P3	Consultant	10
E3	P4	Engineer	48
E4	P2	Programmer	18
E5	P2	Manager	24
E6	P4	Manager	48
E7	P3	Engineer	36

Answer

Given the workson table and the following query, how many rows are returned?

```
SELECT resp, pno, SUM(hours)
FROM workson
WHERE hours > 10
GROUP BY resp, pno
```

A) 9 B) 7 C) 5 D) 1 E) 0



Try It: GROUP BY

Example 5.12

Use GROUP BY and aggregation functions to answer these queries.

1. Output the total number of projects in the database.
2. Return the sum of the budgets for all projects.
3. For each department (dno), return the department number (dno) and the average budget of projects in that department.
4. For each project (pno), return the project number (pno) and the sum of the number of hours employees have worked on that project.
   - Challenge: Show the project name (pname) as well as the project number.
5. Challenge: Show the department name (dname), project name (pname), and sum of hours worked on that project as well as the number of employees working on the project.

Putting it all together

The steps to write an English query in SQL are:

1. Find the columns that you need and put in `SELECT` clause.
2. List the tables that have the columns in the `FROM` clause. If there is more than one, join them together (using `JOIN`/`ON`)
3. If you must filter rows, add a filter criteria in `WHERE` clause.
4. If you need to create an aggregate, use aggregation functions (e.g. `COUNT`, `AVG`) and `GROUP BY`.
5. If you must filter aggregates, add a filter criteria in a `HAVING` clause.

Putting it all together

Example: For each project name list the sum of the hours worked by employees working as a 'Manager' on the project.

```
SELECT pname, SUM(hours) as totalHours
FROM workson INNER JOIN proj on workson.pno=proj.pno
WHERE resp='Manager'
GROUP BY pname
```

Microsoft Access Querying Summary

1. Projection is performed by selecting the fields in the output in the field row in the table at the bottom of the screen.
2. Selection is performed by entering the condition in the criteria box. The criteria applies to the field in that column.
3. The tables used are added to the query by the Show Table. . . option.
4. Joins (based on relationships) are often automatically added, but if not, you can add them by selecting the join field in one table, holding the mouse button, then dragging to the join field in the other table.

Conclusion

A **database** is a collection of related data. A database system allows storing and querying a database.

SQL is the standard query language for databases, although Microsoft Access also provides a graphical user interface.

`CREATE TABLE` creates a table. `INSERT, DELETE`, and `UPDATE` commands modify the data stored within the database.

The basic query operations are selection (subset of rows), projection (subset of columns), join (combine two or more tables), and grouping and aggregation.

Objectives

- Define: database, database system, schema, metadata
- Define: relation, attribute, tuple, domain, degree, cardinality
- SQL properties: reserved words, case-insensitive, free-format
- Be able to create a table using CREATE TABLE command and in Microsoft Access.
- Explain what a key is and what it is used for.
- Use DROP TABLE to delete a table and its data.
- Use INSERT/UPDATE/DELETE to add/update/delete rows of a table and perform same actions using Microsoft Access user interface.
- Execute queries using SQL SELECT and using Microsoft Access user interface.
- Sort rows using ORDER BY. Use LIMIT to keep only the first (top) N rows.
- Use GROUP BY and aggregation functions for calculating summary data.
- Given a small database write simple English queries in SQL.