Databases, Part II: Retrieving Information

In this section you will learn about how information can be retrieved via queries.

Online MS-Office information source: https://support.office.com/

Reminder: Purpose Of A Database

- To **retrieve** information information
Database Queries

• Queries are questions ‘asked’ of/to the database in order to retrieve information.

Retrieving Data Via Queries

• Data retrieval occurs through the use of ‘queries’:
  – As mentioned: A query is a question asked of the data in the database.
  – May be worded to show only the parts of the database for which the answer to the question is true OR it be worded to just show the values of the attributes specified in the query
  – Example: What is the CallSign, FirstName, LastName and Level of gamers in the Gamers Table:

Query (this graphical form is an Access specific)
Forming Queries Graphically In Access

• Create->Query Design

Forming Queries Graphically In Access (2)

• Select a table or tables (whose attributes will be displayed in the query)
Forming Queries Graphically In Access (3)

- Select the **attributes** of the table

Forming Queries Graphically In Access (4)

- Run the query to view the results: Design->Run
Query #1: Specifying Query Criteria

• What is the CallSign & Email & and FirstName, LastName of all gamers with the last name of Tam?

<table>
<thead>
<tr>
<th>CallSign</th>
<th>Email</th>
<th>Telephone</th>
<th>Income</th>
<th>LastName</th>
<th>FirstName</th>
<th>Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>a123</td>
<td><a href="mailto:foe@bar.ca">foe@bar.ca</a></td>
<td></td>
<td>$12,000,000,000.00</td>
<td>Carter</td>
<td>Mary</td>
<td>L9</td>
</tr>
<tr>
<td>Amazing</td>
<td></td>
<td></td>
<td>$0.00</td>
<td></td>
<td></td>
<td>L05</td>
</tr>
<tr>
<td>Cowboy</td>
<td>countryboy@hot (111)11111111</td>
<td></td>
<td>$123,000.00</td>
<td>Texan</td>
<td>Tough</td>
<td>L9</td>
</tr>
<tr>
<td>Fonz</td>
<td></td>
<td></td>
<td>$42,500.00</td>
<td>Edgar</td>
<td>Maureen</td>
<td>L1</td>
</tr>
<tr>
<td>Freeloader</td>
<td><a href="mailto:cheap@skate.or">cheap@skate.or</a></td>
<td></td>
<td>$0.00</td>
<td>You gotta be...kidding me!</td>
<td>L3</td>
<td></td>
</tr>
<tr>
<td>Maverick</td>
<td><a href="mailto:rabel@yell.ca">rabel@yell.ca</a></td>
<td></td>
<td>$75,000.00</td>
<td>Maverick</td>
<td>John</td>
<td>L7</td>
</tr>
<tr>
<td>RestV1</td>
<td></td>
<td></td>
<td>$35,000.00</td>
<td>Kendrey</td>
<td>Leon</td>
<td>L3</td>
</tr>
<tr>
<td>RestV2</td>
<td></td>
<td></td>
<td>$42,000.00</td>
<td>Redfield</td>
<td>Claire</td>
<td>L5</td>
</tr>
<tr>
<td>sss775</td>
<td></td>
<td></td>
<td>$0.00</td>
<td>Jones</td>
<td>Mary</td>
<td>L2</td>
</tr>
<tr>
<td>SilentHtl</td>
<td>heather@morris</td>
<td></td>
<td>$6,500.00</td>
<td>Maurice</td>
<td>Heather</td>
<td>L7</td>
</tr>
<tr>
<td>SilentMtn</td>
<td><a href="mailto:harry@gmason.o">harry@gmason.o</a></td>
<td></td>
<td>$55,000.00</td>
<td>Mecoon</td>
<td>Harri</td>
<td>L4</td>
</tr>
<tr>
<td>Slayer</td>
<td><a href="mailto:tam_yea_right@hotmail.com">tam_yea_right@hotmail.com</a></td>
<td></td>
<td>$100,000.00</td>
<td>Tam</td>
<td>James</td>
<td>L8</td>
</tr>
<tr>
<td>2Miley</td>
<td><a href="mailto:1@PL.com">1@PL.com</a></td>
<td></td>
<td>$1.00</td>
<td>Wang</td>
<td>Tam</td>
<td>L7</td>
</tr>
<tr>
<td>Tamman</td>
<td><a href="mailto:tama@aol.com">tama@aol.com</a></td>
<td></td>
<td>$55,000.00</td>
<td>Tam</td>
<td>Tam</td>
<td>L2</td>
</tr>
<tr>
<td>Tomstone</td>
<td><a href="mailto:gm.all@gmail.co">gm.all@gmail.co</a></td>
<td></td>
<td>$25,000.00</td>
<td>Torrie</td>
<td>Donald</td>
<td>L02</td>
</tr>
</tbody>
</table>

SQL (Structured Query Language)

• It’s the universal language for querying a database (very widely used!)
• Unlike graphical queries the statements can be used in different database programs (not just Access)
• Queries are formed using text descriptions (can be more powerful but more complex than graphical queries):
  – SELECT: Specifies the relevant attributes of which tables are involved in the query e.g., CallSign attribute of the Gamers table
  – FROM: Lists the tables from which the data is to be selected
  – WHERE: Provides the conditions to determine if a particular row shows or doesn’t show as a query result
  – ORDER BY: Specifies the order in which rows are to be returned;

Note: Capitalizing of the above four words is a standard SQL convention.
Simple SQL Queries: Generic Format

• SELECT: \(<Table \ name^1>.<Attribute \ name^1>, <Table \ name^1>.<Attribute \ name^1>...<Table \ name^1>.<Attribute \ name^1>\)
• FROM: \(<Table \ name^1>, <Table \ name^1>...<Table \ name^1>\)

1. Only include the tables and attributes that will actually be displayed in the query results

Example SQL Query

• Example: What is the CallSign, FirstName, LastName and Level of every gamer in the Gamers Table:
• Graphical Access query:

  ![Graphical Access Query](image)

• SQL:
  ```sql
  SELECT Gamers.CallSign, Gamers.FirstName, Gamers.LastName, Gamers.Level
  FROM Gamers;
  ```

Note: not all attributes specified in the query
CPSC 203: What You Need To Know About Forming Queries

• You need to know how to form and evaluate/trace queries (determine what will appear when the query is run) graphically (“Query Design”) or using SQL for both the assignment and for the examination component.

Query #1B: Using An Attribute In Query Condition But Not Displayed As A Result

• Previous Example:
  – What is the CallSign & Email & and FirstName, LastName of all gamers with last name of Tam?

<table>
<thead>
<tr>
<th>CallSign</th>
<th>Email</th>
<th>FirstName</th>
<th>LastName</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slayer</td>
<td><a href="mailto:tam_yeah_right@hotmail.com">tam_yeah_right@hotmail.com</a></td>
<td>James</td>
<td>Tam</td>
</tr>
<tr>
<td>Tamman</td>
<td><a href="mailto:tama@aol.com">tama@aol.com</a></td>
<td>Tam</td>
<td>Tam</td>
</tr>
</tbody>
</table>

  – I know all gamers displayed in the results will have ‘Tam’ for a last name (why display obvious information)
Calculated Values

• In Access they are attributes /columns of the query that are derived from one or more attributes/columns of the tables used in the query

GAMES Table

<table>
<thead>
<tr>
<th>Title</th>
<th>RatePerMinute</th>
</tr>
</thead>
<tbody>
<tr>
<td>TheTams</td>
<td>$0.33</td>
</tr>
</tbody>
</table>

(In the query this value is derived from the hourly rate)
RatePerMinute = HourlyRate / 60

Specifying Calculated Values (Access Specific)

• Graphically (MS-Access via DesignView)

Format:
<Query column name>:
<Attribute name> or constant <expression>
<Attribute name> or constant <expression>...
<Attribute name> or constant

Example:
RatePerMinute: [HourlyRate] / 60
Specifying **Calculated Values** (SQL)

Format:
```
SELECT <Table name.Attribute name³>,
      <Table name.Attribute name²>, ... 
      <![Attribute name] or constant> <expression>
      <![Attribute name] or constant> <expression> ...
      <![Attribute name] or constant> AS <Query column name>
      <Table name.Attribute nameⁿ>
```

Example:
• SELECT Games.Title, Games.HourlyRate, [HourlyRate]/60 
  AS RatePerMinute

---

Query #2: Calculated Values

• Show the Title, HourlyRate, RatePerMinute of the games in our system.
  – Note:
    • A more useful and realistic query would calculate the cost of each gaming 
      session as a gamer plays a game.
    • That would involve querying multiple tables (Games and Sessions) which is 
      more complex so that query will be formed later in this section of notes.

• Back to the basic query: A conversion is needed
  – Cost for playing a game is cost/hour
  – Calculation needed for the missing information: 
    
    CostPerMinute = CostPerHour / 60

---
Query # 2: Graphical Access Query (DesignView)

- **Calculated Values** were used: The results of some columns were **not stored** in tables but derived only as the query is run from the values in other columns.

<table>
<thead>
<tr>
<th>RatePerMinute</th>
<th>RatePerMinute</th>
<th>RatePerMinute</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="RatePerMinute" /></td>
<td><img src="image2" alt="RatePerMinute" /></td>
<td><img src="image3" alt="RatePerMinute" /></td>
</tr>
</tbody>
</table>

**RatePerMinute:**

\[
\frac{\text{HourlyRate}}{60}
\]

Query #2: SQL Form And The Graphical DesignView

- SELECT Games.Title, Games.HourlyRate, \([\text{HourlyRate}] / 60\) AS RatePerMinute
- FROM Gamers;

![Query SQL Form](image4)
Access: Cleaning Up The Results

• Select the attribute

• Ribbon: Select the design tab,
  – Select the property sheet option

Logical Operators

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
<th>MS-Access operator</th>
</tr>
</thead>
</table>
| AND       | • All conditions must be true for the result to be true.  
            • If any condition is false then the entire result is false. | And |
| OR        | • All conditions must be false for the result to be false.  
            • If any condition is true then the entire result is true. | Or |
Relational Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;</td>
<td>Less than</td>
</tr>
<tr>
<td>&lt;=</td>
<td>Less than, equal to</td>
</tr>
<tr>
<td>&gt;</td>
<td>Greater than</td>
</tr>
<tr>
<td>&gt;=</td>
<td>Greater than, equal to</td>
</tr>
<tr>
<td>=</td>
<td>Equal to</td>
</tr>
<tr>
<td>&lt;&gt;</td>
<td>Not equal to</td>
</tr>
</tbody>
</table>

SQL: The ‘Where’ Clause

- **SELECT**: Specifies the relevant attributes of which tables are involved in the query e.g., CallSign attribute of the Gamers table
- **FROM**: Lists the tables from which the data is to be selected
- **WHERE**: Provides the conditions to determine if a particular row appears in the query results

**Format**

WHERE: `<Boolean expression> <logic operator> <Boolean expression>`

**Examples**:

WHERE: Client.Age >= 0 AND Client.Age <= 114
WHERE: Client.LastName = "Tam"
Query #3: Logical-OR

- Show the CallSign, LastName & FirstName of all employees whose last name is “Maurice” or “Masoon”.

<table>
<thead>
<tr>
<th>CallSign</th>
<th>LastName</th>
<th>FirstName</th>
</tr>
</thead>
<tbody>
<tr>
<td>SilentMtn</td>
<td>Masoon</td>
<td>Harry</td>
</tr>
<tr>
<td>SilentHL</td>
<td>Maurice</td>
<td>Heather</td>
</tr>
</tbody>
</table>

- Note that query criteria specified within a column will have the logical OR operation applied.

Query #4: Logical-AND

- Show all online sessions of the game title “DOOMED” which had a session lasting an hour or more.

<table>
<thead>
<tr>
<th>Title</th>
<th>SessionDuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>DOOMED</td>
<td>60</td>
</tr>
<tr>
<td>DOOMED</td>
<td>1200</td>
</tr>
<tr>
<td>DOOMED</td>
<td>300</td>
</tr>
<tr>
<td>DOOMED</td>
<td>360</td>
</tr>
<tr>
<td>DOOMED</td>
<td>12000</td>
</tr>
</tbody>
</table>

- Note that query criteria specified between columns will have the logical AND operation applied to them.
Ordering Queries

• Query results can be ranked according to the attributes of a table.

<table>
<thead>
<tr>
<th>LastName</th>
<th>FirstName</th>
<th>CallSign</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aamazing</td>
<td>Az</td>
<td>zeephyr</td>
</tr>
<tr>
<td>Carswell</td>
<td>Mary</td>
<td>a123</td>
</tr>
<tr>
<td>Edgar</td>
<td>Maureen</td>
<td>FooS</td>
</tr>
<tr>
<td>Jones</td>
<td>Mary</td>
<td>s1x775</td>
</tr>
<tr>
<td>Kednne</td>
<td>Leon</td>
<td>ResEv1</td>
</tr>
<tr>
<td>Mason</td>
<td>Harry</td>
<td>SilentMtn</td>
</tr>
<tr>
<td>Maurice</td>
<td>Heather</td>
<td>SilentHl</td>
</tr>
<tr>
<td>Maverick</td>
<td>John</td>
<td>Maverick</td>
</tr>
<tr>
<td>Redfield</td>
<td>Claire</td>
<td>ResEv2</td>
</tr>
<tr>
<td>Tam</td>
<td>James</td>
<td>Slayer</td>
</tr>
<tr>
<td>Tam</td>
<td>Tam</td>
<td>Tamman</td>
</tr>
<tr>
<td>Texan</td>
<td>Tough</td>
<td>Cowboy</td>
</tr>
<tr>
<td>Torrie</td>
<td>Donald</td>
<td>Tomstone</td>
</tr>
<tr>
<td>Wang</td>
<td>Tam</td>
<td>SMILey</td>
</tr>
<tr>
<td>You gotta be...</td>
<td>kidding me!</td>
<td>Freeloader</td>
</tr>
</tbody>
</table>

Query #5: Ordering Queries

• Show the CallSign, LastName and FirstName of all the gamers in ascending order sorted first by last name, then by first name and finally by call sign.

• Use the “SORT” property (MS-Access) “ORDER BY” property (SQL example is on the next slide)
**Ordering Queries: SQL Format**

Generic format:

```
ORDER BY <Table name>.<Attribute name>, <Table name>.<Attribute name> ... <Table name>.<Attribute name>
```

Example:
```
ORDER BY Students.LastName, Students.FirstName, Students.MiddleName
```

1 Only include attributes under the ‘ORDER BY’ clause that are involved in the ranking or ordering of query results

Example (rank only according by last name in this example)
```
SELECT Students.LastName, Students.FirstName, Students.MiddleName
From Students
ORDER BY Students.LastName;
```

---

**Query #6A: Contradictory Conditions**

- Take care not to specify queries that can never be true!
- Example:
  - Show the CallSign and Income of all gamers with an income $50,000 or lower AND $100,000 or higher.

```
SELECT Gamers.CallSign, Gamers.Income
FROM Gamers
WHERE (Gamers.Income >= 100000 And Gamers.Income <= 50000);
```
Query 6B: Queries That Are Always True

• Example:
  – What if the previous example to exclude gamers with a last name of “Tam” as well as “Edgar” used the logical-OR operator instead of the logical-AND operator.

```
SELECT Gamers.LastName, Gamers.FirstName
FROM Gamers
WHERE
  (Gamers.LastName <> "Tam") Or
  (Gamers.LastName <> "Edgar");
```

Results:

Queries 7{A-D}: Using The **Wildcard** In Queries

• Similar to validation rules, the wild card can be used in queries when only partial information is known.

• Example: last name schwar*

• Use the wildcard in conjunction with the ‘Like’ operator under ‘Criteria’ (MS-Access) or ‘Where’ & Like (SQL)
**Wild Card: Example Queries**

- Show last name when `LastName` begins with ‘t’
  
  ```
  SELECT Gamers.LastName
  FROM Gamers
  WHERE
  (Gamers.LastName Like "t\*"); 
  ```

- Show first and last name when `FirstName` ends with ‘m’
  
  ```
  SELECT Gamers.FirstName, Gamers.LastName
  FROM Gamers
  WHERE
  (Gamers.FirstName Like "*m"); 
  ```

**Wild Card: Example Queries (2)**

- Show the call sign when there’s an ‘a’ anywhere (call sign).
  
  ```
  SELECT Gamers.CallSign
  FROM Gamers
  WHERE
  (Gamers.CallSign Like "a*"); 
  ```
### Single Character Wildcard ‘?’

- Show .com emails with only a single character between the ‘@’ and the ‘.com’
- (Any number of characters before the at-sign)

<table>
<thead>
<tr>
<th>Table to query:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Email</td>
</tr>
<tr>
<td><a href="mailto:foc@ber.ca">foc@ber.ca</a></td>
</tr>
<tr>
<td>ab.com</td>
</tr>
<tr>
<td><a href="mailto:countryboy@hotmail.com">countryboy@hotmail.com</a></td>
</tr>
<tr>
<td><a href="mailto:cheap@skate.org">cheap@skate.org</a></td>
</tr>
<tr>
<td><a href="mailto:rebel@yell.ca">rebel@yell.ca</a></td>
</tr>
<tr>
<td><a href="mailto:heather@morris.com">heather@morris.com</a></td>
</tr>
<tr>
<td><a href="mailto:harry@mason.com">harry@mason.com</a></td>
</tr>
<tr>
<td><a href="mailto:tam_yeighth@hotmail.com">tam_yeighth@hotmail.com</a></td>
</tr>
<tr>
<td><a href="mailto:1@1.com">1@1.com</a></td>
</tr>
<tr>
<td><a href="mailto:tam@ail.com">tam@ail.com</a></td>
</tr>
<tr>
<td>gm <a href="mailto:ail@gmail.com">ail@gmail.com</a></td>
</tr>
<tr>
<td><a href="mailto:1@.com">1@.com</a></td>
</tr>
</tbody>
</table>

#### Query design

- **Field:** Email
- **Table:** Gamers
- **Sort:**
- **Show:** Like ‘@?.com’

```sql
SELECT Gamers.Email
FROM Gamers
WHERE
    (Gamers.Email Like "@?\..com");
```

#### Query results:

<table>
<thead>
<tr>
<th>Email</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="mailto:a@b.com">a@b.com</a></td>
</tr>
<tr>
<td><a href="mailto:1@1.com">1@1.com</a></td>
</tr>
<tr>
<td><a href="mailto:1@.com">1@.com</a></td>
</tr>
</tbody>
</table>

### Input Masks: The Slash And The Query

- There’s a separate database example to illustrate: “Extra database - input mask and queries”
- Recall: input mask in the design view

- In the Datasheet (data entry) view the ‘A’ character is displayed with the rest of the attribute data
Exported Table: Excel Spreadsheet

- As can be seen the actual data (easiest to see when the database is exported to Excel) the ‘A’ following the slash is not actually stored (‘Age2’ attribute)

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ID</td>
<td>Age1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>A12</td>
</tr>
</tbody>
</table>

Query Results: age1

- The data in ‘age1’ can be queried with ‘a’ is part of the attribute

DesignView of query

Query results: filtering to show values in attribute ‘Age1’ that start with ‘A’
Query Results: age2

- The data in ‘age2’ DOES NOT include ‘a’ as part of the attribute.
- Querying for this data ‘a’ will produce no results (i.e. proof that no values in attribute ‘Age2’ contain a leading ‘A’)

Design View of query:

<table>
<thead>
<tr>
<th>Field</th>
<th>Table</th>
<th>Age1</th>
<th>Age2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Table1</td>
<td>Table1</td>
<td>Table1</td>
</tr>
<tr>
<td>Sort:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Show:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Criteria:</td>
<td></td>
<td>Like &quot;A&quot;</td>
<td></td>
</tr>
</tbody>
</table>

Query results:

Queries Can Span Multiple Tables

- This is referred to as a ‘join’ because the results join data from multiple tables
- Generic format of multi-table queries
  - SELECT: <Table name>..<Attribute name>, <Table name>..<Attribute name>...
  - FROM: <Table name> INNER JOIN <Table name>...INNER JOIN <Table name> ON <Table name>..<Primary key> = <Table name>..<Foreign key>... <Table name>..<Primary key> = <Table name>..<Foreign key>
Query #8: Simple Multi-Table Query

• What is the: Title (Games), HourlyRate (Games) and Date (Sessions) of games that were actually played.
  – A gamer must have played at least one game (created a game session)

<table>
<thead>
<tr>
<th>CallSign</th>
<th>SessionDate</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMiLey</td>
<td>9/13/2015</td>
<td>DOOMED</td>
</tr>
<tr>
<td>Tamman</td>
<td>9/13/2015</td>
<td>TheTams</td>
</tr>
<tr>
<td>a123</td>
<td>9/13/2015</td>
<td>EpicLegends</td>
</tr>
<tr>
<td>Cowboy</td>
<td>9/13/2015</td>
<td>DOOMED</td>
</tr>
<tr>
<td>Tomstone</td>
<td>9/16/2015</td>
<td>DOOMED</td>
</tr>
<tr>
<td>Tomstone</td>
<td>9/14/2015</td>
<td>DOOMED</td>
</tr>
<tr>
<td>Tomstone</td>
<td>9/17/2015</td>
<td>DOOMED</td>
</tr>
<tr>
<td>Cowboy</td>
<td>9/14/2015</td>
<td>TheTams</td>
</tr>
<tr>
<td>Freeloader</td>
<td>9/13/2015</td>
<td>DOOMED</td>
</tr>
<tr>
<td>Freeloader</td>
<td>9/14/2015</td>
<td>DOOMED</td>
</tr>
<tr>
<td>Freeloader</td>
<td>9/15/2015</td>
<td>DOOMED</td>
</tr>
<tr>
<td>Freeloader</td>
<td>9/16/2015</td>
<td>DOOMED</td>
</tr>
<tr>
<td>Freeloader</td>
<td>10/31/2015</td>
<td>DOOMED</td>
</tr>
<tr>
<td>Freeloader</td>
<td>11/17/2015</td>
<td>DOOMED</td>
</tr>
</tbody>
</table>

Query #8: Simple Multi-Table Query In Access

• Add the desired tables to the query

• Select the attributes of these tables relevant to the query

<table>
<thead>
<tr>
<th>Field:</th>
<th>Title</th>
<th>HourlyRate</th>
<th>SessionDate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table:</td>
<td>Games</td>
<td>Games</td>
<td>Sessions</td>
</tr>
<tr>
<td>Sort:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Show:</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
</tbody>
</table>
Query #8: SQL

• We will create it ‘live’ in lecture to show you how to form a multi-table query.
  – That is: take notes so you are more likely to learn and remember how to form a multiple table query for the exam.

Query #9: Complex Multi-Table Queries

• What is the: CallSign (Gamers), Income (Gamers), Title (Games) and Date (Sessions) of games that were played by gamers whose income was $100,000 or greater.
  – First restriction: gamers with 6 figure incomes

  • ‘Big money’ gamers that have actually played a game: a123, Cowboy
  • Note: ‘Slayer’ has never played a game
Query #9: Complex Multi-Table Queries (Access)

Forming the query (graphically in Access)

<table>
<thead>
<tr>
<th>Field</th>
<th>CallSign</th>
<th>Income</th>
<th>Title</th>
<th>SessionDate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table</td>
<td>Gamers</td>
<td>Gamers</td>
<td>Games</td>
<td>Sessions</td>
</tr>
<tr>
<td>Sort</td>
<td>Show:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Criteria</td>
<td></td>
<td>$\geq$ 100000</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Query results

<table>
<thead>
<tr>
<th>CallSign</th>
<th>Income</th>
<th>Title</th>
<th>SessionDate</th>
</tr>
</thead>
<tbody>
<tr>
<td>a123</td>
<td>$12,000.00</td>
<td>EpicLegends</td>
<td>9/1/2015</td>
</tr>
<tr>
<td>Cowboy</td>
<td>$123,000.00</td>
<td>DOOMED</td>
<td>9/1/2015</td>
</tr>
<tr>
<td>Cowboy</td>
<td>$123,000.00</td>
<td>Frankenstein'sHorror</td>
<td>2/9/2015</td>
</tr>
<tr>
<td>Cowboy</td>
<td>$123,000.00</td>
<td>TheTims</td>
<td>9/14/2015</td>
</tr>
</tbody>
</table>

Query #9: Complex Multi-Table Queries (Building SQL With Reduced Bracketing)

```sql
SELECT Gamers.CallSign, Gamers.Income, Games.Title, Sessions.SessionDate
FROM Games INNER JOIN Gamers
INNER JOIN Sessions
ON Gamers.CallSign = Sessions.CallSign
WHERE Gamers.Income $\geq$ 100000;
```
Query #10: Putting Multiple Things Together (Multi-Table Query Using Calculated Values)

• Remember an earlier query that introduced calculated values (simple but not very practical)
  – SELECT Games.Title, sGames.HourlyRate, [HourlyRate]/60
    AS RatePerMinute

• The new query illustrates the following concepts: calculated values, multi-table queries.
  – Show the Title (Games), SessionDate (Sessions), HourlyRate (Games), RatePerMinute, SessionDuration (Sessions) and the SessionCost of games that were played.

Again Recall Query #2: Graphical Access Query

• “Calculated Values”: The results of some columns were not stored in tables but derived from the values in other columns

<table>
<thead>
<tr>
<th>Field</th>
<th>Table</th>
<th>Sort</th>
<th>Show</th>
<th>Criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>Games</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SessionDate</td>
<td>Sessions</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HourlyRate</td>
<td>Games</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RatePerMinute</td>
<td>HourlyRate, 60</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Making This query more useful/realistic (multiple tables required)

RatePerMinute: [HourlyRate] / 60

SessionCost: [RatePerMinute] * [SessionDuration]
Query 10: **Multi-Table Query With Calculated Values**

- **SELECT** Games.Title, Sessions.SessionDate, Games.HourlyRate, 
  
  \[\frac{\text{HourlyRate}}{60}\] AS RatePerMinute, Sessions.SessionDuration, 
  
  \[\text{RatePerMinute} \times \text{SessionDuration}\] AS SessionCost

- **FROM** Games INNER JOIN (Gamers INNER JOIN Sessions ON 
  Gamers.CallSign = Sessions.CallSign) ON Games.Title = 
  Sessions.Title;

**Calculated Values:** **Result Is Always Stored** In Column 
Of A Table (Attribute)

- Deriving values only as a query is run vs. deriving a value and 
  storing the value in a column of a table.
  
- Deriving a value as a column value:
  
  - The result of the calculation **applies to every instance/row** in the table
  
  - General example: Employees table, the amount that the worker pays into the 
    company pension plan depends upon how much the employee earns and is 
    generally applicable.
  
  - Example from the database: 
    
    \[\text{RatePerMinute} = \frac{\text{HourlyRate}}{60}\]
    
    - \text{RatePerMinute} is an attribute of every game
    
    - \text{RatePerMinute} And \text{HourlyRate} are both attributes from the same table.
Calculated Values: **Result Is Calculated Only When A Query Runs**

– Deriving a value only as a query is run:
  
  • The information should not be stored in the table: the *result only applies to some instances/rows* only and/or it would be a waste of space to store it in the table.
  
  • General example: Employees table, cost of paying the worker an early retirement buyout is often offered only to older employees who are already close to retirement.
  
  • Example from the database:
    
    \[
    \text{SessionCost} = \text{RatePerMinute} \times \text{SessionDuration}
    \]

    – Is SessionCost a logical attribute of a game?
    – Is SessionCost a logical attribute of a session?
    – MS-Access won’t allow SessionCost as a table attribute because it involves attributes from multiple tables: RatePerMinute is an attribute of the Games table while SessionDuration is an attribute of the Sessions table.
    – For this reason SessionCost cannot be a calculated value of a table and can only be a calculated value derived during a query.

**NOTE:** Look carefully at your assignment requirements in order to determine if any calculated values must be derived only during the query. You get few (if any) marks by storing a value in a table that is to be derived only during the query.

---

**Query #10: Forming The Queries**

• SELECT Games.Title, Sessions.SessionDate, Games.HourlyRate, \([\text{HourlyRate}] / 60\) AS RatePerMinute, Sessions.SessionDuration, \([\text{RatePerMinute}] \times [\text{SessionDuration}]\) AS SessionCost

• FROM Games INNER JOIN Sessions ON Games.Title = Sessions.Title;
After This Section You Should Now Know

• How to form different queries in order to retrieve data from a database
  • Specifying which attributes will appear
  • Specifying which tables to include
  • Specify query conditions (Boolean expressions often using logical-AND, logical-OR and sometimes logical-NOT or inequality)
  • Ordering queries
  • Specifying query results from multiple tables
• How to derive values: calculated values either stored as an attribute (column of a table) or derived only during a query
• How malformed queries can either take the form of:
  • Contradictions (always false) result in no query results
  • Always true queries results in every instance displayed as a query result

After This Section You Should Now Know (2)

• How wildcards (single and multi-character) can be used in queries
• **How to form and trace** (predict the result of) queries specified graphically (“Query Design” method of MS-Access) or the more general use **SQL format**.