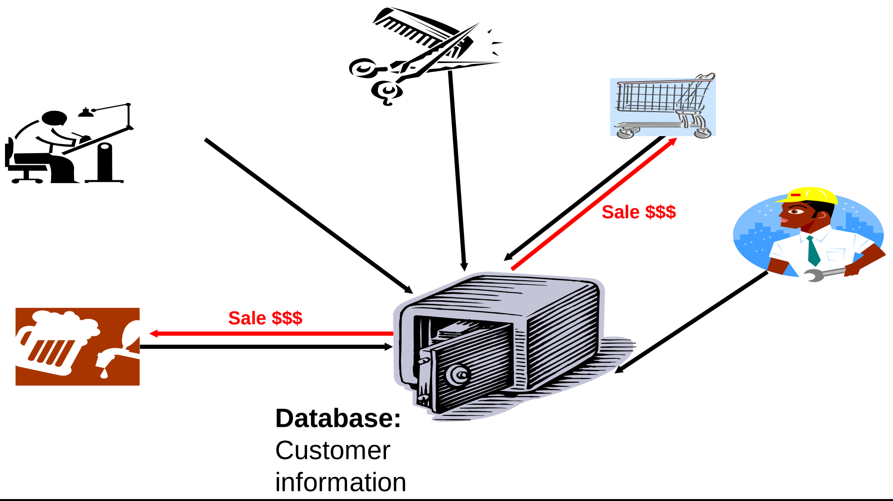
Databases, Part II: Retrieving Information

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

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

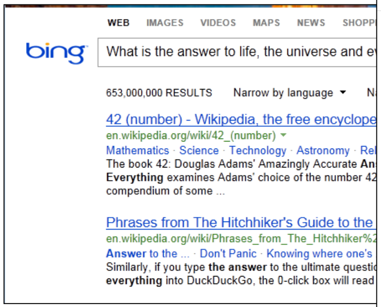
Reminder: Purpose Of A Database

- To **retrieve** information information



Database Queries

- Queries are questions 'asked' of/to the database in order to retrieve information.



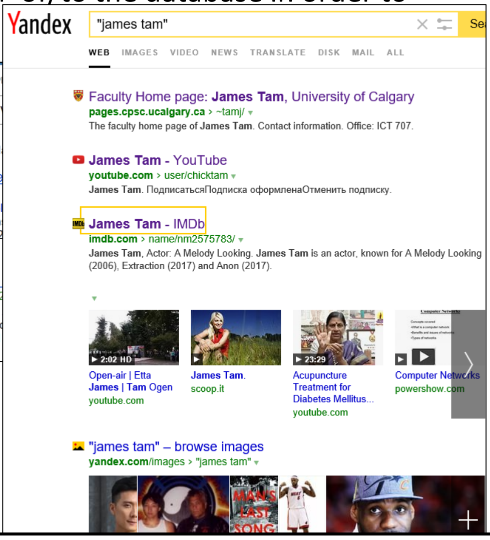
WEB IMAGES VIDEOS MAPS NEWS SHOPP

bing What is the answer to life, the universe and e

653,000,000 RESULTS Narrow by language ▾ N

42 (number) - Wikipedia, the free encyclope
en.wikipedia.org/wiki/42_(number) ▾
Mathematics · Science · Technology · Astronomy · Rel
The book 42: Douglas Adams' Amazingly Accurate **An**
Everything examines Adams' choice of the number 42
compendium of some ...

Phrases from The Hitchhiker's Guide to the
en.wikipedia.org/wiki/Phrases_from_The_Hitchhiker%2
Answer to the ... · Don't Panic · Knowing where one's
Similarly, if you type **the answer** to the ultimate questi
everything into DuckDuckGo, the 0-click box will read



Yandex "james tam"

WEB IMAGES VIDEO NEWS TRANSLATE DISK MAIL ALL

Faculty Home page: **James Tam**, University of Calgary
pages.cpsc.ucalgary.ca › ~tamj/ ▾
The faculty home page of **James Tam**. Contact information. Office: ICT 707.

James Tam - YouTube
youtube.com › user/chicktam ▾
James Tam. ПодписатьсяПодписка оформленаОтменить подписку.

James Tam - IMDb
imdb.com › name/nm2575783/ ▾
James Tam, Actor: A Melody Looking. **James Tam** is an actor, known for A Melody Looking (2006), Extraction (2017) and Anon (2017).

2:02 HD Open-air | Etta **James** | **Tam** Ogen youtube.com
James Tam. scoop.it
23:29 Acupuncture Treatment for Diabetes Mellitus... youtube.com
Computer Networks powershow.com

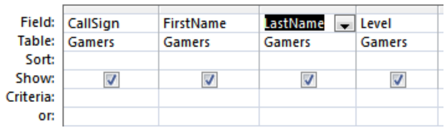
"james tam" – browse images
yandex.com/images › "james tam" ▾

"What is the answer to life, the universe and everything"
– *The Hitchhiker's Guide to the Galaxy* (Del Rey 1979)
– *The Hitchhiker's Guide to the Galaxy* (BBC 1981)

Retrieving Data Via Queries

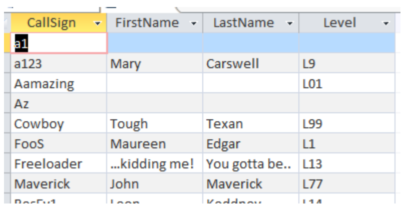
- Data retrieval occurs through the use of 'queries':
  - As mentioned: A query is a question asked of the data in the database.
  - May be worded to show only the parts of the database for which the answer to the question is true OR it be worded to just show the values of the attributes specified in 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)



Field:	CallSign	FirstName	LastName	Level
Table:	Gamers	Gamers	Gamers	Gamers
Sort:				
Show:	✓	✓	✓	✓
Criteria:				
or:				

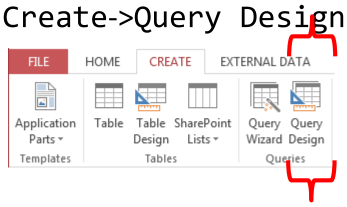
Result of the query



CallSign	FirstName	LastName	Level
a1			
a123	Mary	Carswell	L9
Aamazing			L01
Az			
Cowboy	Tough	Texan	L99
FooS	Maureen	Edgar	L1
Freeloader	...kidding me!	You gotta be..	L13
Maverick	John	Maverick	L77

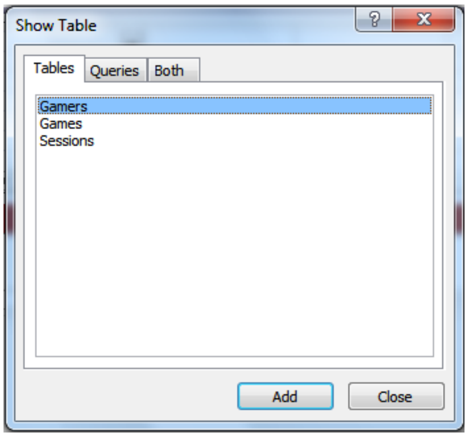
Forming Queries Graphically In Access

- `Create->Query Design`



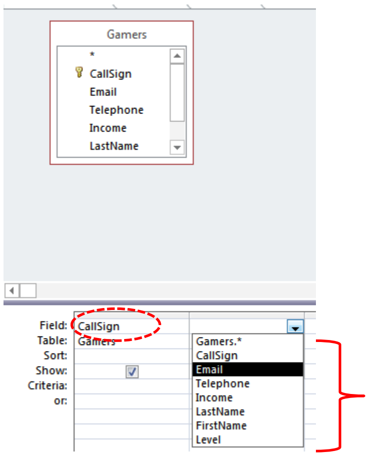
Forming Queries Graphically In Access (2)

- Select a table or tables (whose attributes will be displayed in the query)



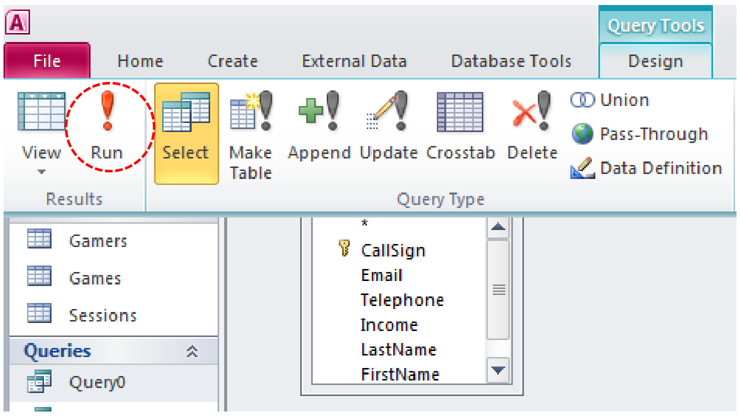
Forming Queries Graphically In Access (3)

- Select the **attributes** of the table



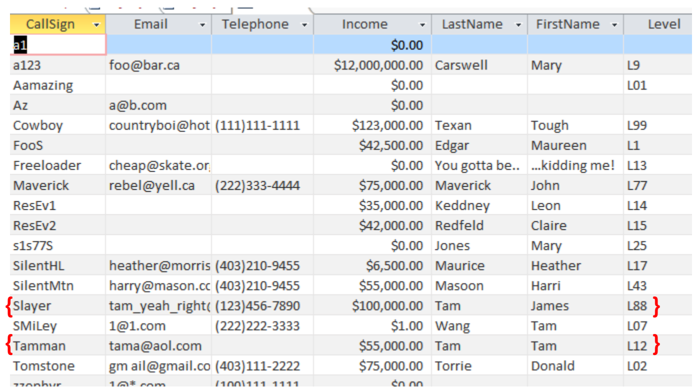
Forming Queries Graphically In Access (4)

- Run the query to view the results: `Design`->Run

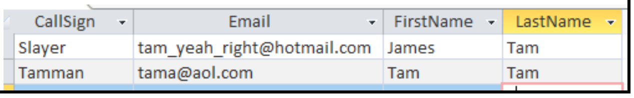


Query #1: Specifying Query Criteria

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



CallSign	Email	Telephone	Income	LastName	FirstName	Level
a1			$0.00			
a123	foo@bar.ca		$12,000,000.00	Carswell	Mary	L9
Aamazing			$0.00			L01
Az	a@b.com		$0.00			
Cowboy	countryboi@hot	(111)111-1111	$123,000.00	Texan	Tough	L99
FooS			$42,500.00	Edgar	Maureen	L1
Freeloader	cheap@skate.or		$0.00	You gotta be..	...kidding me!	L13
Maverick	rebel@yell.ca	(222)333-4444	$75,000.00	Maverick	John	L77
ResEv1			$35,000.00	Keddney	Leon	L14
ResEv2			$42,000.00	Redfeld	Claire	L15
s1s77S			$0.00	Jones	Mary	L25
SilentHL	heather@morris	(403)210-9455	$6,500.00	Maurice	Heather	L17
SilentMtn	harry@mason.cc	(403)210-9455	$55,000.00	Masoon	Harri	L43
{Slayer	tam_yeah_right(	(123)456-7890	$100,000.00	Tam	James	L88 }
SMiLey	1@1.com	(222)222-3333	$1.00	Wang	Tam	L07
{Tamman	tama@aol.com		$55,000.00	Tam	Tam	L12 }
Tomstone	gm ail@gmail.co	(403)111-2222	$75,000.00	Torrie	Donald	L02
zzephyr	1@*.com	(100)111-1111	$0.00			



CallSign	Email	FirstName	LastName
Slayer	tam_yeah_right@hotmail.com	James	Tam
Tamman	tama@aol.com	Tam	Tam

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.

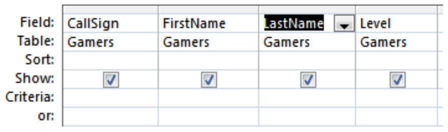
Simple SQL Queries: Generic Format

- `SELECT: <Table name1>.<Attribute name1>, <Table name1>.<Attribute name1>….`
- `FROM: <Table name1>, <Table name1>… <Table name1>`

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

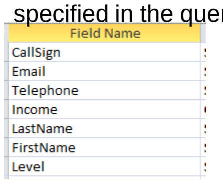
Example SQL Query

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



Field:	CallSign	FirstName	LastName	Level
Table:	Gamers	Gamers	Gamers	Gamers
Sort:				
Show:	✓	✓	✓	✓
Criteria:				
or:				

Note: not all attributes specified in the query



Field Name
CallSign
Email
Telephone
Income
LastName
FirstName
Level

- **SQL:**

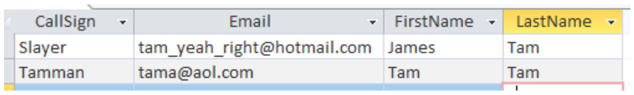
```
SELECT Gamers.CallSign, Gamers.FirstName,
       Gamers.LastName, Gamers.Level
FROM Gamers;
```

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.

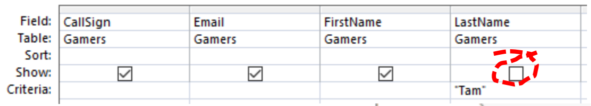
Query #1B: Using An Attribute In Query Condition But **Not Displayed** As A Result

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

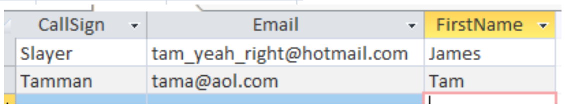


CallSign	Email	FirstName	LastName
Slayer	tam_yeah_right@hotmail.com	James	Tam
Tamman	tama@aol.com	Tam	Tam

  - I know all gamers displayed in the results will have 'Tam' for a last name (why display obvious information)



Field:	CallSign	Email	FirstName	LastName
Table:	Gamers	Gamers	Gamers	Gamers
Sort:				
Show:	☑	☑	☑	☐
Criteria:				"Tam"



CallSign	Email	FirstName
Slayer	tam_yeah_right@hotmail.com	James
Tamman	tama@aol.com	Tam

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

Title	RatePerMinute
TheTams	$0.33

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

Specifying Calculated Values (Access Specific)

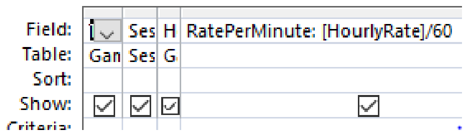
- Graphically (MS-Access via `DesignView`)

Format:

<Query column name>:
<[Attribute name] or constant> <expression>
<[Attribute name] or constant> <expression>...
<[Attribute name] or constant>

Example:

RatePerMinute: [HourlyRate] / 60



Field:		Ses	H	RatePerMinute: [HourlyRate]/60
Table:	Gan	Ses	G	
Sort:				
Show:	☑	☑	☑	☑
Criteria:				

Specifying **Calculated Values** (SQL)

Format:

```
SELECT <Table name.Attribute name1>,
<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:

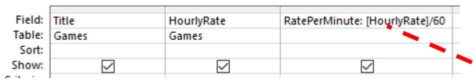
- `SELECT Games.Title, Games.HourlyRate, [HourlyRate]/60 AS RatePerMinute`

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.

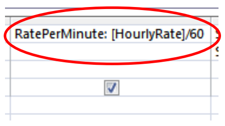


Field:	Title	HourlyRate	RatePerMinute: [HourlyRate]/60
Table:	Games	Games	
Sort:			
Show:	☑	☑	☑

```
RatePerMinute:
[HourlyRate] / 60
```

Query #2: SQL Form And The Graphical DesignView

- `SELECT Games.Title, Games.HourlyRate, [HourlyRate]/60 AS RatePerMinute`
- `FROM Gamers;`

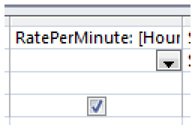


RatePerMinute: [HourlyRate]/60

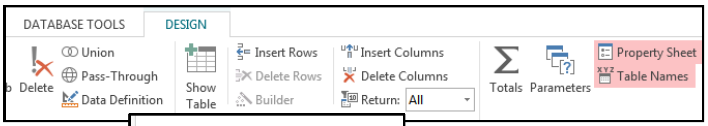
☑

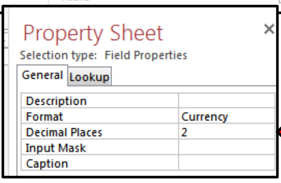
Access: Cleaning Up The Results

- Select the attribute

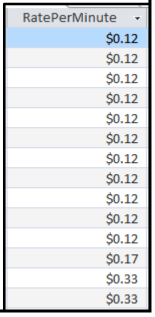

RatePerMinute: [Hour

- Ribbon: Select the design tab,
  - Select the property sheet option




Property Sheet
Selection type: Field Properties
General | Lookup

Description	
Format	Currency
Decimal Places	2
Input Mask	
Caption	



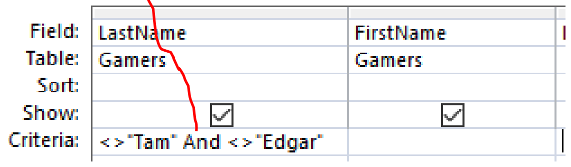
RatePerMinute
$0.12
$0.12
$0.12
$0.12
$0.12
$0.12
$0.12
$0.12
$0.12
$0.12
$0.12
$0.17
$0.33
$0.33

Logical Operators

Operation	Description	MS-Access operator
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

Operator	Description
<	Less than
<=	Less than , equal to
>	Greater than
>=	Greater than, equal to
=	Equal to
<>	Not equal to



Field:	LastName	FirstName
Table:	Gamers	Gamers
Sort:		
Show:	✓	✓
Criteria:	<>"Tam" And <>"Edgar"	

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**

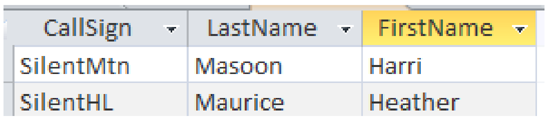
```
WHERE: <Boolean expression> <logic operator>
       <Boolean expression>
```

- **Examples**:

```
WHERE: Client.Age >= 0 AND Client.Age <= 114
WHERE: Client.LastName = "Tam"
```

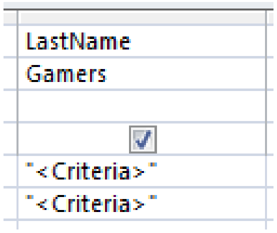
Query #3: Logical-OR

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



CallSign	LastName	FirstName
SilentMtn	Masoon	Harri
SilentHL	Maurice	Heather

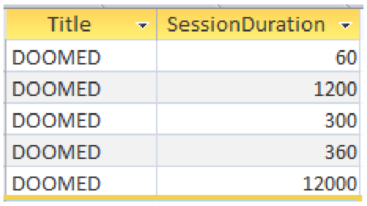
- Note that query criteria specified *within a column* will have the logical OR operation applied.



LastName
Gamers
✓
"<Criteria>"
"<Criteria>"

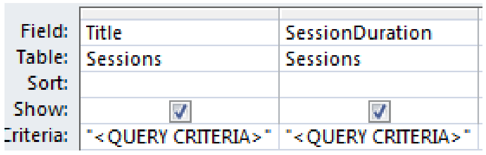
Query #4: Logical-AND

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



Title	SessionDuration
DOOMED	60
DOOMED	1200
DOOMED	300
DOOMED	360
DOOMED	12000

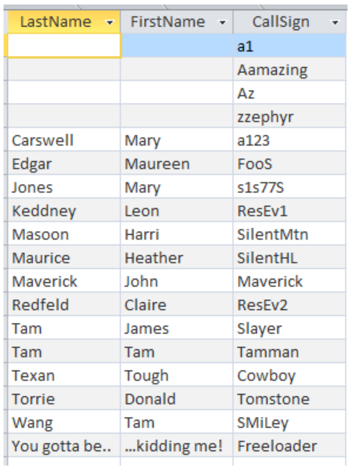
- Note that query criteria specified *between columns* will have the logical AND operation applied to them.



Field:	Title	SessionDuration
Table:	Sessions	Sessions
Sort:		
Show:	✓	✓
Criteria:	"<QUERY CRITERIA>"	"<QUERY CRITERIA>"

Ordering Queries

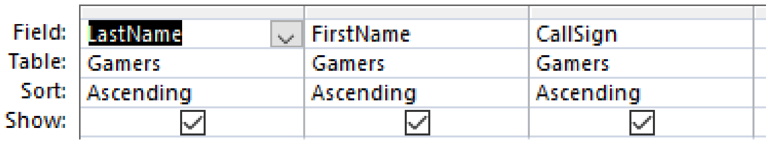
- Query results can be ranked according to the attributes of a table.



LastName	FirstName	CallSign
		a1
		Aamazing
		Az
		zzephyr
Carswell	Mary	a123
Edgar	Maureen	FooS
Jones	Mary	s1s77S
Keddney	Leon	ResEv1
Masoon	Harri	SilentMtn
Maurice	Heather	SilentHL
Maverick	John	Maverick
Redfeld	Claire	ResEv2
Tam	James	Slayer
Tam	Tam	Tamman
Texan	Tough	Cowboy
Torrie	Donald	Tomstone
Wang	Tam	SMiLey
You gotta be..	...kidding me!	Freeloader

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)



Field:	LastName	FirstName	CallSign
Table:	Gamers	Gamers	Gamers
Sort:	Ascending	Ascending	Ascending
Show:	✓	✓	✓

Ordering Queries: SQL Format

Generic format:

ORDER BY *<Table name[1]>.<Attribute name[1]>, <Table name[1]>.<Attribute name[1]>... <Table name[1]>.<Attribute name[1]>*

Example:

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

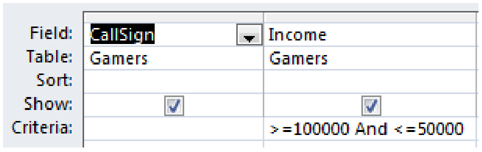
1 Only include attributes under the 'ORDER BY' clause that are involved in the ranking or ordering of query results

Example (rank only according by last name in this example)

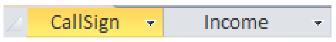
```
SELECT Students.LastName, Students.FirstName, Students.MiddleName
From Students
ORDER BY: Students.LastName;
```

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.



Field:	CallSign	Income
Table:	Gamers	Gamers
Sort:		
Show:	✓	✓
Criteria:		>=100000 And <=50000

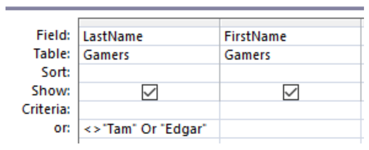


CallSign	Income

```
SELECT Gamers.CallSign,
Gamers.Income
FROM Gamers
WHERE (Gamers.Income >=100000 And
              Gamers.Income
       <=50000);
```

Query 6B: Queries That Are Always True

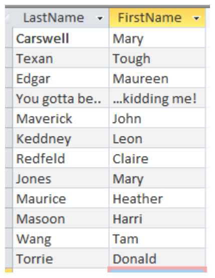
- Example:
  - What if the previous example to exclude gamers with a last name of "Tam" as well as "Edgar" used the logical-OR operator instead of the logical-AND operator.



Field:	LastName	FirstName
Table:	Gamers	Gamers
Sort:		
Show:	✓	✓
Criteria:		
or:	<>"Tam" Or "Edgar"	

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

Results:



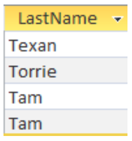
LastName	FirstName
Carswell	Mary
Texan	Tough
Edgar	Maureen
You gotta be..	...kidding me!
Maverick	John
Keddney	Leon
Redfeld	Claire
Jones	Mary
Maurice	Heather
Masoon	Harri
Wang	Tam
Torrie	Donald

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

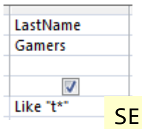
- Similar to validation rules, the wild card can be used in queries when only partial information is known.
- Example: last name schwar*
- Use the wildcard in conjunction with the 'Like' operator under 'Criteria' (MS-Access) or 'Where' & Like (SQL)

Wild Card: Example Queries

- Show last name when `LastName` begins with 't'



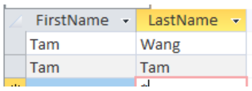
LastName
Texan
Torrie
Tam
Tam



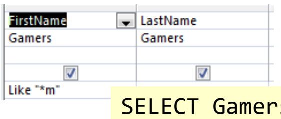
LastName
Gamers
✓
Like "t*"

```
SELECT Gamers.LastName
FROM Gamers
WHERE
  (Gamers.LastName Like "t*");
```

- Show first and last name when `FirstName` ends with 'm'



FirstName	LastName
Tam	Wang
Tam	Tam

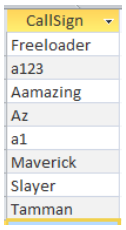


FirstName	LastName
Gamers	Gamers
✓	✓
Like "*m"	

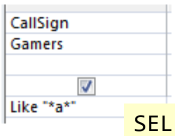
```
SELECT Gamers.FirstName,
Gamers.LastName
FROM Gamers
WHERE
   (Gamers.FirstName Like "*m");
```

Wild Card: Example Queries (2)

- Show the call sign when there's an 'a' anywhere (call sign).



CallSign
Freeloader
a123
Aamazing
Az
a1
Maverick
Slayer
Tamman



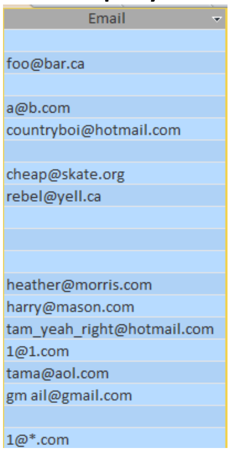
CallSign
Gamers
✓
Like "*a*"

```
SELECT Gamers.CallSign
FROM Gamers
WHERE
  (Gamers.CallSign Like "*a*");
```

Single Character Wildcard '?'

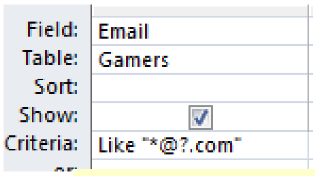
- Show .com emails with only a single character between the '@' and the '.com'
- (Any number of characters before the at-sign)

Table to query:



Email
foo@bar.ca
a@b.com
countryboi@hotmail.com
cheap@skate.org
rebel@yell.ca
heather@morris.com
harry@mason.com
tam_yeah_right@hotmail.com
1@1.com
tama@aol.com
gm ail@gmail.com
1@*.com

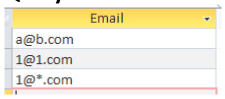
Query design



Field:	Email
Table:	Gamers
Sort:	
Show:	✓
Criteria:	Like "*@?.com"

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

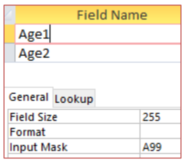
Query results:



Email
a@b.com
1@1.com
1@*.com

Input Masks: The Slash And The Query

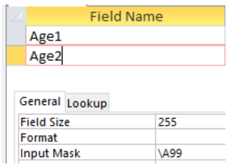
- There's a separate database example to illustrate: "`Extra database - input mask and queries`"
- Recall: input mask in the design view



Field Name
Age1
Age2

General | Lookup

Field Size	255
Format	
Input Mask	A99

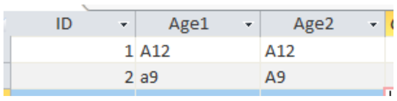


Field Name
Age1
Age2

General | Lookup

Field Size	255
Format	
Input Mask	\A99

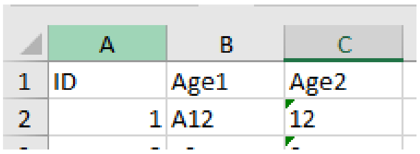
- In the Datasheet (data entry) view the 'A' character is displayed with the rest of the attribute data



ID	Age1	Age2
1	A12	A12
2	a9	A9

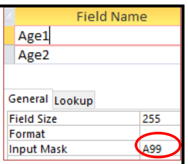
Exported Table: Excel Spreadsheet

- 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)



	A	B	C
1	ID	Age1	Age2
2	1	A12	12

Query Results: age1



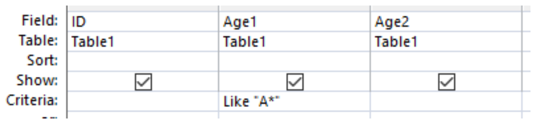
Field Name
Age1
Age2

General Lookup

Field Size	255
Format	
Input Mask	A99

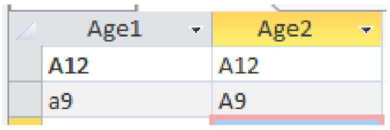
- The data in 'age1' can be queried with 'a' is part of the attribute

DesignView of query



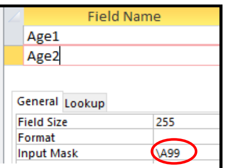
Field:	ID	Age1	Age2
Table:	Table1	Table1	Table1
Sort:			
Show:	☑	☑	☑
Criteria:		Like "A*"	

Query results: filtering to show values in attribute 'Age1' that start with 'A'



Age1	Age2
A12	A12
a9	A9

Query Results: age2

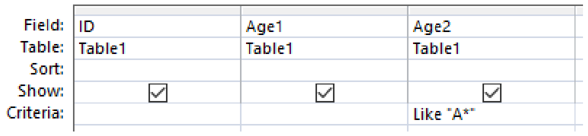


Field Name
Age1
Age2

Field Size	255
Format	
Input Mask	\A99

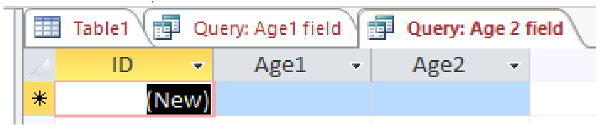
- The data in ‘age2’ DOES NOT include ‘a’ as part of the attribute.
- Querying for this data ‘a’ will produce no results (i.e. proof that no values in attribute ‘Age2’ contain a leading ‘A’)

DesignView of query



Field:	ID	Age1	Age2
Table:	Table1	Table1	Table1
Sort:			
Show:	✓	✓	✓
Criteria:			Like "A*"

Query results



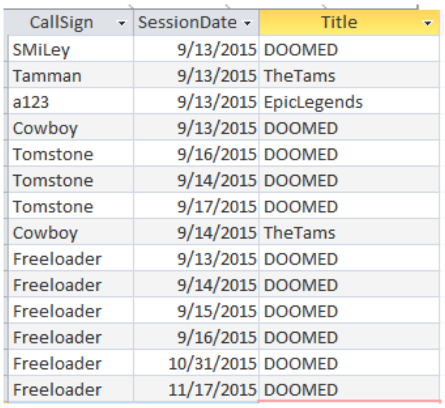
ID	Age1	Age2
(New)		

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

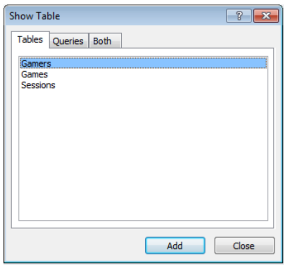
- What is the: `Title` (Games), `HourlyRate` (Games) and `Date` (`Sessions`) of games that were actually played.
  – A gamer must have played at least one game (created a game session)



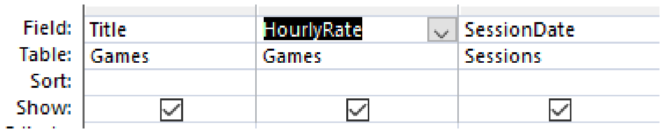
CallSign	SessionDate	Title
SMiLey	9/13/2015	DOOMED
Tamman	9/13/2015	TheTams
a123	9/13/2015	EpicLegends
Cowboy	9/13/2015	DOOMED
Tomstone	9/16/2015	DOOMED
Tomstone	9/14/2015	DOOMED
Tomstone	9/17/2015	DOOMED
Cowboy	9/14/2015	TheTams
Freeloader	9/13/2015	DOOMED
Freeloader	9/14/2015	DOOMED
Freeloader	9/15/2015	DOOMED
Freeloader	9/16/2015	DOOMED
Freeloader	10/31/2015	DOOMED
Freeloader	11/17/2015	DOOMED

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



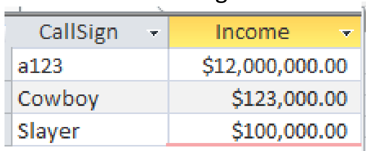
Field:	Title	HourlyRate	SessionDate
Table:	Games	Games	Sessions
Sort:			
Show:	✓	✓	✓

Query #8: SQL

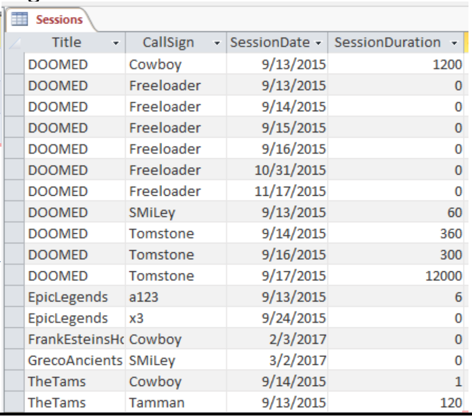
- We will create it 'live' in lecture to show you how to form a multi-table query.
  - That is: take notes so you are more likely to learn and remember how to form a multiple table query for the exam.

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



CallSign	Income
a123	$12,000,000.00
Cowboy	$123,000.00
Slayer	$100,000.00



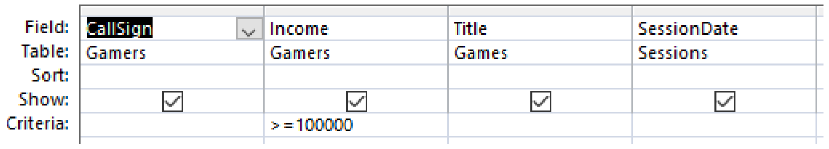
Sessions

Title	CallSign	SessionDate	SessionDuration
DOOMED	Cowboy	9/13/2015	1200
DOOMED	Freeloader	9/13/2015	0
DOOMED	Freeloader	9/14/2015	0
DOOMED	Freeloader	9/15/2015	0
DOOMED	Freeloader	9/16/2015	0
DOOMED	Freeloader	10/31/2015	0
DOOMED	Freeloader	11/17/2015	0
DOOMED	SMiLey	9/13/2015	60
DOOMED	Tomstone	9/14/2015	360
DOOMED	Tomstone	9/16/2015	300
DOOMED	Tomstone	9/17/2015	12000
EpicLegends	a123	9/13/2015	6
EpicLegends	x3	9/24/2015	0
FrankEsteinsHc	Cowboy	2/3/2017	0
GrecoAncients	SMiLey	3/2/2017	0
TheTams	Cowboy	9/14/2015	1
TheTams	Tamman	9/13/2015	120

- 'Big money' gamers that have actually played a game: a123, Cowboy
- Note: 'Slayer' has never played a game

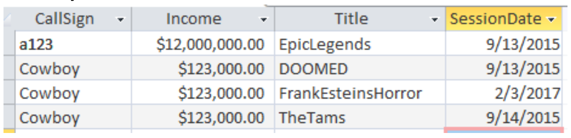
Query #9: Complex Multi-Table Queries (Access)

Forming the query
(graphically in Access)



Field:	CallSign	Income	Title	SessionDate
Table:	Gamers	Gamers	Games	Sessions
Sort:				
Show:	✓	✓	✓	✓
Criteria:		>=100000		

Query results



CallSign	Income	Title	SessionDate
a123	$12,000,000.00	EpicLegends	9/13/2015
Cowboy	$123,000.00	DOOMED	9/13/2015
Cowboy	$123,000.00	FrankEsteinsHorror	2/3/2017
Cowboy	$123,000.00	TheTams	9/14/2015

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

```
SELECT Gamers.CallSign, Gamers.Income, Games.Title,
Sessions.SessionDate
```

```
FROM: <Table name> INNER
JOIN <Table name>…

…ON Table name>.<Primary
key> = <Table
name>.<Foreign key>...
<Table name>.<Primary key>
= <Table name>.<Foreign
key>
```

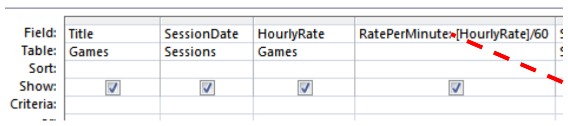
```
FROM Games INNER JOIN Gamers
INNER JOIN Sessions

ON Gamers.CallSign =
Sessions.CallSign)

ON Games.Title =
Sessions.Title


WHERE Gamers.Income >=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

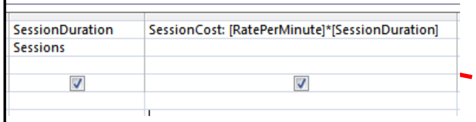
- "**Calculated Values**": The results of some columns were not stored in tables but derived from the values in other columns



Field:	Title	SessionDate	HourlyRate	RatePerMinute: [HourlyRate]/60	S
Table:	Games	Sessions	Games		S
Sort:					
Show:	✓	✓	✓	✓	
Criteria:					

```
RatePerMinute:
[HourlyRate] / 60
```

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



SessionDuration	SessionCost: [RatePerMinute]*[SessionDuration]
Sessions	
✓	✓

```
SessionCost:
[RatePerMinute] * [SessionDuration]
```

Query 10: **Multi-Table Query** With **Calculated Values**

- `SELECT Games.Title, Sessions.SessionDate, Games.HourlyRate, [HourlyRate]/60 AS RatePerMinute, Sessions.SessionDuration, [RatePerMinute]*[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: `RatePerMinute = HourlyRate / 60`
      - `RatePerMinute` is an attribute of every game
      - `RatePerMinute` And `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:

```
SessionCost = RatePerMinute * SessionDuration
```

  – Is `SessionCost` a logical attribute of a game?

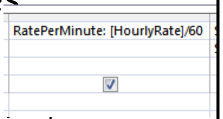
  – Is `SessionCost` a logical attribute of a session?

  – 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

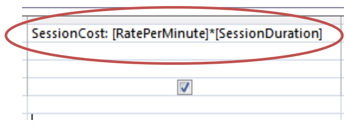
Query #10: Forming The Queries



RatePerMinute: [HourlyRate]/60

✓

- `SELECT Games.Title, Sessions.SessionDate, Games.HourlyRate, [HourlyRate]/60 AS RatePerMinute, Sessions.SessionDuration, [RatePerMinute]*[SessionDuration] AS SessionCost`



SessionCost: [RatePerMinute]*[SessionDuration]

✓

- `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**.