Data Mining In Modern Astronomy Sky Surveys:  

*Databases*  
& *Sloan Digital Sky Survey*  

Ching-Wa Yip  

cwyip@pha.jhu.edu; Bloomberg 518
Discussion of HW2:
Estimation of Hubble’s Constant

R Console

> # Read Hubble data file
> data <- read.csv('H:/public_html/teaching/hubbletable1.csv')
> # Show data
> data

<table>
<thead>
<tr>
<th>ObjectName</th>
<th>Distance_Mpc</th>
<th>RecessionVelocity_kms</th>
</tr>
</thead>
<tbody>
<tr>
<td>SmallMag</td>
<td>0.032</td>
<td>170</td>
</tr>
<tr>
<td>LargeMag</td>
<td>0.034</td>
<td>290</td>
</tr>
<tr>
<td>NGC6822</td>
<td>0.214</td>
<td>-130</td>
</tr>
<tr>
<td>NGC598</td>
<td>0.263</td>
<td>-70</td>
</tr>
<tr>
<td>NGC221</td>
<td>0.275</td>
<td>-185</td>
</tr>
<tr>
<td>NGC224</td>
<td>0.275</td>
<td>-220</td>
</tr>
<tr>
<td>NGC5457</td>
<td>0.450</td>
<td>200</td>
</tr>
<tr>
<td>NGC4736</td>
<td>0.500</td>
<td>290</td>
</tr>
<tr>
<td>NGC5194</td>
<td>0.500</td>
<td>270</td>
</tr>
<tr>
<td>NGC4449</td>
<td>0.630</td>
<td>200</td>
</tr>
<tr>
<td>NGC4214</td>
<td>0.800</td>
<td>300</td>
</tr>
<tr>
<td>NGC3031</td>
<td>0.900</td>
<td>-30</td>
</tr>
<tr>
<td>NGC3627</td>
<td>0.900</td>
<td>650</td>
</tr>
<tr>
<td>NGC4826</td>
<td>0.900</td>
<td>150</td>
</tr>
<tr>
<td>NGC5236</td>
<td>0.900</td>
<td>500</td>
</tr>
<tr>
<td>NGC1068</td>
<td>1.000</td>
<td>920</td>
</tr>
<tr>
<td>NGC5055</td>
<td>1.100</td>
<td>450</td>
</tr>
<tr>
<td>NGC7331</td>
<td>1.100</td>
<td>500</td>
</tr>
<tr>
<td>NGC4258</td>
<td>1.400</td>
<td>500</td>
</tr>
<tr>
<td>NGC4151</td>
<td>1.700</td>
<td>960</td>
</tr>
<tr>
<td>NGC4382</td>
<td>2.000</td>
<td>500</td>
</tr>
<tr>
<td>NGC4472</td>
<td>2.000</td>
<td>850</td>
</tr>
<tr>
<td>NGC4486</td>
<td>2.000</td>
<td>800</td>
</tr>
<tr>
<td>NGC4649</td>
<td>2.000</td>
<td>1090</td>
</tr>
</tbody>
</table>
> # Plot Recession Velocity vs. Distance of galaxies
> plot(data$Distance_Mpc, data$RecessionVelocity_kms)
> # Fit linear model
> fit1 <- lm(data$RecessionVelocity_kms ~ data$Distance_Mpc)
> # Add best-fit straight line
> abline(fit1, col = 'red')
> # Show best-fit parameters
> fit1

Call:
lm(formula = data$RecessionVelocity_kms ~ data$Distance_Mpc)

Coefficients:
 (Intercept)   data$Distance_Mpc
       -40.78          454.16

> # Fit origin-passing linear model
> fit2 <- lm(data$RecessionVelocity_kms ~ data$Distance_Mpc + 0)
> # Add best-fit straight line
> abline(fit2, col = 'blue')
> # Show best-fit parameters
> fit2
Red: $y = mx + c$
Blue: $y = mx$ (i.e., $c = 0$ or origin passing)
Discussion HW2

- The calculated values (454 km/s/Mpc) is a factor of a few larger than the WMAP value (71 km/s/Mpc).
- This discrepancy suggests that there could be systematic error in Hubble’s measurements of Recession Velocity or/and Distance. The error could be due to the measurement techniques and/or the local galaxy sample.
- Photon count = 100 implies SNR = \sqrt{100} = 10.
- By using the simplified Rose Criterion, the minimum number of photons for 100% feature detection is 5^2 = 25.
Further Readings on Data Mining and Machine Learning

• **Statistical Data Analysis** (Cowan)
  – Practical reference/textbook
• **A Modern Introduction to Probability and Statistics** (Dekking, Kraaikamp, Lopuhaä, Meester)
  – Self-content textbook
  – Freely downloadable online
• **All of Statistics** (Wasserman)
  – Comprehensive; Advanced read
• **Neural Networks for Pattern Recognition** (Bishop)
  – Focus on concepts
  – Freely downloadable online
From Data to Information

• We don’t just want data.
• We want information from the data.

Information

Database

Sensors

Data Analysis
or
Data Mining
Topics

• Database
• Table
• Structured Query Language (SQL)
• Sloan Digital Sky Survey (SDSS) and Web Services
• Example SQL queries in Astronomy:
  – Create binned histograms of galaxies
  – Select targets for follow-up spectroscopy
  – Find extreme galaxies (i.e., outliers)
Basics of Database

- A database stores a collection of data.
- The data are arranged in database objects such as tables.
- Relational Database: a database which uses table(s).
  - The “relation” refers to the relation among different fields within one table.
  - The “relation” does not refer to the potential relation among multiple tables.
Basics of Tables

• Row is called **Record**.
• Column is called **Field**.
• Schema: logical container for database objects that user creates.
• Records are stored in the tables with some order:
  – The records are not necessarily sorted by a particular column.
Table contains Unique Records: Primary Key

- We want to be able to retrieve each and every record.
- Solution: Each record in a table is unique.
- This unique ID is called **Primary Key**.
- In the SDSS, some Primary Keys are:
  - ObjID (in table PhotoObjAll)
  - SpecObjID (in table SpecObjAll)

<table>
<thead>
<tr>
<th>First Name</th>
<th>Last Name</th>
<th>Credit Card #</th>
</tr>
</thead>
<tbody>
<tr>
<td>George</td>
<td>Daniels</td>
<td>184715170968</td>
</tr>
<tr>
<td>Amy</td>
<td>Lee</td>
<td>207609796702</td>
</tr>
<tr>
<td>Brandon</td>
<td>Willis</td>
<td>982767757110</td>
</tr>
<tr>
<td>Jennifer</td>
<td>Connolly</td>
<td>486830981903</td>
</tr>
<tr>
<td>Andrew</td>
<td>Folks</td>
<td>601571389801</td>
</tr>
</tbody>
</table>
Table contains Unique Records: Primary Key

• We want to be able to retrieve each and every record.
• Solution: Each record in a table is unique.
• This unique ID is called Primary Key.
• In the SDSS, some Primary Keys are:
  – ObjID (in table PhotoObjAll)
  – SpecObjID (in table SpecObjAll)

<table>
<thead>
<tr>
<th>ID</th>
<th>First Name</th>
<th>Last Name</th>
<th>Credit Card #</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>George</td>
<td>Daniels</td>
<td>184715170968</td>
</tr>
<tr>
<td>2</td>
<td>Amy</td>
<td>Lee</td>
<td>207609796702</td>
</tr>
<tr>
<td>3</td>
<td>Brandon</td>
<td>Willis</td>
<td>982767757110</td>
</tr>
<tr>
<td>4</td>
<td>Jennifer</td>
<td>Connolly</td>
<td>486830981903</td>
</tr>
<tr>
<td>5</td>
<td>Andrew</td>
<td>Folks</td>
<td>601571389801</td>
</tr>
</tbody>
</table>
Foreign Key

• A Foreign Key is a field of a table (*child table*) that uniquely identifies a row in another table (*parent table*).
• A Foreign Key hence ties two tables together.
• In the “Customer and Purchase” tables, CreditCard # is the Foreign Key.
Un-Normalized Table

• In un-normalized table:
  – Records may grow very quickly.
  – Redundant records may present.
• Solution: Split data into multiple tables.
• In Astronomy: Data are fixed once the survey is completed. But tables are long, normalization improves performance.
• In Industry (banking/searching/facebook etc.): Data are growing fast, giving many records for a given user. Normalization is important.
Un-Normalized Table:
Purchase

<table>
<thead>
<tr>
<th>First Name</th>
<th>Last Name</th>
<th>Credit Card #</th>
<th>Date</th>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>George</td>
<td>Daniels</td>
<td>184715170968</td>
<td>01/05/2013</td>
<td>125.6</td>
</tr>
<tr>
<td>Amy</td>
<td>Lee</td>
<td>207609796702</td>
<td>01/07/2013</td>
<td>45.50</td>
</tr>
<tr>
<td>George</td>
<td>Daniels</td>
<td>184715170968</td>
<td>01/07/2013</td>
<td>72.35</td>
</tr>
<tr>
<td>Brandon</td>
<td>Willis</td>
<td>982767757110</td>
<td>01/09/2013</td>
<td>38.97</td>
</tr>
<tr>
<td>Jennifer</td>
<td>Connolly</td>
<td>486830981903</td>
<td>01/08/2013</td>
<td>49.83</td>
</tr>
<tr>
<td>George</td>
<td>Daniels</td>
<td>184715170968</td>
<td>01/10/2013</td>
<td>72.35</td>
</tr>
<tr>
<td>Andrew</td>
<td>Folks</td>
<td>601571389801</td>
<td>01/12/2013</td>
<td>92.30</td>
</tr>
</tbody>
</table>

- There are redundant data in this table.
Split Data into 2 Tables: Customer and Purchase

<table>
<thead>
<tr>
<th>First Name</th>
<th>Last Name</th>
<th>Credit Card #</th>
</tr>
</thead>
<tbody>
<tr>
<td>George</td>
<td>Daniels</td>
<td>184715170968</td>
</tr>
<tr>
<td>Amy</td>
<td>Lee</td>
<td>207609796702</td>
</tr>
<tr>
<td>Brandon</td>
<td>Willis</td>
<td>982767757110</td>
</tr>
<tr>
<td>Jennifer</td>
<td>Connolly</td>
<td>486830981903</td>
</tr>
<tr>
<td>Andrew</td>
<td>Folks</td>
<td>601571389801</td>
</tr>
</tbody>
</table>

- No redundant data.
- Two tables grow at different rate!

<table>
<thead>
<tr>
<th>Credit Card #</th>
<th>Date</th>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>184715170968</td>
<td>01/05/2013</td>
<td>125.6</td>
</tr>
<tr>
<td>207609796702</td>
<td>01/07/2013</td>
<td>45.50</td>
</tr>
<tr>
<td>184715170968</td>
<td>01/07/2013</td>
<td>72.35</td>
</tr>
<tr>
<td>982767757110</td>
<td>01/09/2013</td>
<td>38.97</td>
</tr>
<tr>
<td>486830981903</td>
<td>01/08/2013</td>
<td>49.83</td>
</tr>
<tr>
<td>184715170968</td>
<td>01/10/2013</td>
<td>72.35</td>
</tr>
<tr>
<td>601571389801</td>
<td>01/12/2013</td>
<td>92.30</td>
</tr>
</tbody>
</table>
### Split Data into 2 Tables: Customer and Purchase

<table>
<thead>
<tr>
<th>First Name</th>
<th>Last Name</th>
<th>Credit Card #</th>
</tr>
</thead>
<tbody>
<tr>
<td>George</td>
<td>Daniels</td>
<td>184715170968</td>
</tr>
<tr>
<td>Amy</td>
<td>Lee</td>
<td>207609796702</td>
</tr>
<tr>
<td>Brandon</td>
<td>Willis</td>
<td>982767757110</td>
</tr>
<tr>
<td>Jennifer</td>
<td>Connolly</td>
<td>486830981903</td>
</tr>
<tr>
<td>Andrew</td>
<td>Folks</td>
<td>601571389801</td>
</tr>
</tbody>
</table>

- No redundant data.
- Two tables grow at different rate!

A single spreadsheet is not the best approach for storing big data!

<table>
<thead>
<tr>
<th>Credit Card #</th>
<th>Date</th>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>184715170968</td>
<td>01/05/2013</td>
<td>125.6</td>
</tr>
<tr>
<td>207609796702</td>
<td>01/07/2013</td>
<td>45.50</td>
</tr>
<tr>
<td>184715170968</td>
<td>01/07/2013</td>
<td>72.35</td>
</tr>
<tr>
<td>982767757110</td>
<td>01/09/2013</td>
<td>38.97</td>
</tr>
<tr>
<td>486830981903</td>
<td>01/08/2013</td>
<td>49.83</td>
</tr>
<tr>
<td>184715170968</td>
<td>01/10/2013</td>
<td>72.35</td>
</tr>
<tr>
<td>601571389801</td>
<td>01/12/2013</td>
<td>92.30</td>
</tr>
</tbody>
</table>
85 Tables in SDSS DR7
Sloan Digital Sky Survey (2000-)

- Photometric + Spectroscopic Surveys
  - 11,000 square degree footprint (DR7)
  - $5.9 \times 10^8$ u, g, r, i, z photometry
  - $1.6 \times 10^6$ fiber spectra
- Phases
  - SDSS I (2000-05)
  - SDSS II (2005-08)
  - SDSS III (2008-14)
  - SDSS 4 (Current)
- Data are public
- Web interfaces for data download & exploration
  - SkyServer, DAS, etc.
SDSS Footprints (DR7): in Galactic Coordinate Systems
SDSS Footprints (DR7):
in Galactic Coordinate Systems

Photometry

Spectroscopy

Southern Stripes: Offer repeated scans (time-domain information) of the sky!
SDSS III

• BOSS
  – Map distribution of galaxies out to redshift of 0.7, which has imprints information about the cosmic microwave background.

• SEGUE-2
  – Map Milky Way structure by measuring optical spectra of 119,000 stars.

• APOGEE
  – Map dust-obscured disk and bulge of Milky Way by measuring Infrared spectra of stars.

• MARVELS
  – Search for exoplanets by monitoring radial velocities of 11,000 stars.
Statistics of SDSS Databases (Data Release 7, or DR7)

- Number of tables: 85
- Data Volume:
  - Images (16 TB)
  - Tables (18 TB)
  - Data Products (27 TB)
- PhotoObjAll
  - Number of rows: 585,634,220
  - Number of columns: 454
- SpecObjAll
  - Number of rows: 1,640,960
  - Number of columns: 63
Web Services for SDSS Data

- SkyServer and CasJobs
  - Nolan Li, Alex Szalay, Ani Thakar, Tamas Budavari et al.
- Spectrum Services
  - Dobos et al.
- Open SkyQuery
  - Dobos et al. 2014 in prep.
SkyServer.org - Team

The team behind the skyserver are multitalented and have various backgrounds. You have seen the names - here are the faces.

Tamas Budavari
William O'Mullane

George Fekete
Adrian Pope

Sam Carliles
Jordan Raddick

Nolan Li
Alex Szalay

Maria Nieto-Santisteban
Ani Thakar

Tanu Malik
Jan van den Berg

JHU Intersesion Course - C. W. Yip
Welcome to the **DR7 site!!!**

This website presents data from the Sloan Digital Sky Survey, a project to make a map of a large part of the universe. We would like to show you the beauty of the universe, and share with you our excitement as we build the largest map in the history of the world.

**SkyServer Tools**
- Famous places
- Get images
- Visual Tools
- Explore
- Search
- Object Cross-ID
- CasJobs

**Science Projects**
- Basic
- Advanced
- Challenges
- For Kids
- Games and Contests
- Teachers
- Links to other projects

**Info Links**
- About Astronomy
- About the SDSS
- About the SkyServer
- SDSS Data Release 7
- SDSS Project Website
- Open SkyQuery
- Images of RC3 Galaxies

**Help**
- Getting Started
- FAQ
- How To
- Glossary
- Schema Browser
- Sample SQL Queries
- Details of SDSS Data

**For Astronomers**
- A separate branch of this website for professional astronomers (English)
- More...

**SDSS is supported by**
- [Institution Logos]

**Site Traffic**
- [Traffic Information]

**Contact Us**
- [Contact Information]
### PhotoObjAll

Contains a record describing the attributes of each photometric object.

The table has views:

- **PhotoObj**: all primary and secondary objects; essentially this is the view you should use unless you want a specific type of object.
- **PhotoPrimary**: all photo objects that are primary (the best version of the object).
  - **Star**: Primary objects that are classified as stars.
  - **Galaxy**: Primary objects that are classified as galaxies.
  - **Sky**: Primary objects which are sky samples.
  - **Unknown**: Primary objects which are one of the above.
- **PhotoSecondary**: all photo objects that are secondary (secondary detections).
- **PhotoFamily**: all photo objects which are neither primary nor secondary (blended).

The table has indices that cover the popular columns:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Length</th>
<th>Unit</th>
<th>UCD</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>objID</td>
<td>bigint</td>
<td>8</td>
<td></td>
<td>ID_MAIN</td>
<td>Unique SDSS identifier composed from [skymap, run, camcol, field, obj].</td>
</tr>
<tr>
<td>skyVersion</td>
<td>tinyint</td>
<td>1</td>
<td></td>
<td>CODE_MISC</td>
<td>0 = OPDB target, 1 = OPDB best</td>
</tr>
<tr>
<td>run</td>
<td>smallint</td>
<td>2</td>
<td></td>
<td>OBS_RUN</td>
<td>Run number</td>
</tr>
<tr>
<td>rerun</td>
<td>smallint</td>
<td>2</td>
<td></td>
<td>CODE_MISC</td>
<td>Rerun number</td>
</tr>
<tr>
<td>camcol</td>
<td>tinyint</td>
<td>1</td>
<td></td>
<td>INST_ID</td>
<td>Camera column</td>
</tr>
<tr>
<td>field</td>
<td>smallint</td>
<td>2</td>
<td></td>
<td>ID_FIELD</td>
<td>Field number</td>
</tr>
<tr>
<td>obj</td>
<td>smallint</td>
<td>2</td>
<td></td>
<td>ID_NUMBER</td>
<td>The object id within a field. Usually changes between runs or the same field.</td>
</tr>
<tr>
<td>mode</td>
<td>tinyint</td>
<td>1</td>
<td></td>
<td>CLASS_OBJECT</td>
<td>1: primary, 2: secondary, 3: family object, 4: outside chunk boundary.</td>
</tr>
<tr>
<td>nChild</td>
<td>smallint</td>
<td>2</td>
<td></td>
<td>NUMBER</td>
<td>Number of children if this is a composite object that has been deblended. BRIGHT (in a flags sense) objects also have nchild == 1, the non-BRIGHT sibling.</td>
</tr>
<tr>
<td>type</td>
<td>smallint</td>
<td>2</td>
<td></td>
<td>CLASS_OBJECT</td>
<td>Morphological type classification of the object.</td>
</tr>
<tr>
<td>clean</td>
<td>int</td>
<td>4</td>
<td></td>
<td>CODE_MISC</td>
<td>Clean photometry flag for point sources (1:clean, 0:unclean).</td>
</tr>
<tr>
<td>probPSF</td>
<td>real</td>
<td>4</td>
<td></td>
<td>STAT_PROBABILITY</td>
<td>Probability that the object is a star. Currently 0 if type == 3 (galaxy), 1 if type == 6 (star).</td>
</tr>
<tr>
<td>insideMask</td>
<td>tinyint</td>
<td>1</td>
<td>CODE_MISC</td>
<td>Flag to indicate whether object is inside a mask and why.</td>
<td></td>
</tr>
<tr>
<td>flags</td>
<td>bigint</td>
<td>8</td>
<td>CODE_MISC</td>
<td>Photo Object Attribute Flags.</td>
<td></td>
</tr>
</tbody>
</table>
Using Microsoft SQL Server in Astronomy (Szalay & Gray)

Other choices:
- Oracle
- MySQL
Practical SQL

- We execute queries to manage and retrieve the data.
- The queries are written in Structured Query Language (SQL), which has the form:

```
SELECT column(s) FROM table(s) WHERE predicate(s) are true
```

- SQL queries can get long and complicated.
A Simplest Query: Count Rows

```
SELECT COUNT(*)
FROM THUNDER.BESTD7.dbo.PhotoObjAll
```
### SQL Query

```sql
SELECT TOP 10 
FROM THUMPER.BESTDR7.dbo.PhotoObjAll
```

### Results

<table>
<thead>
<tr>
<th>objID</th>
<th>skyVersion</th>
<th>run</th>
<th>renum</th>
<th>camcol</th>
<th>field</th>
<th>objs</th>
<th>mode</th>
<th>nChild</th>
<th>type</th>
<th>clean</th>
<th>probPSF</th>
<th>insideMask</th>
<th>flags</th>
<th>rowc</th>
<th>rowcErr</th>
<th>colc</th>
<th>co</th>
</tr>
</thead>
<tbody>
<tr>
<td>5877229516333303829</td>
<td>1</td>
<td>745</td>
<td>40</td>
<td>1</td>
<td>518</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>6</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>34825317962</td>
<td>67.31985</td>
<td>0.008972449</td>
<td>49.60184</td>
<td>0</td>
</tr>
<tr>
<td>5877229516333303830</td>
<td>1</td>
<td>745</td>
<td>40</td>
<td>1</td>
<td>518</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>31013825616078</td>
<td>67.25411</td>
<td>0.002276111</td>
<td>49.65848</td>
<td>0</td>
</tr>
<tr>
<td>5877229516333303831</td>
<td>1</td>
<td>745</td>
<td>40</td>
<td>1</td>
<td>518</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>281619127075148</td>
<td>-6.684971</td>
<td>0.1010417</td>
<td>108.149</td>
<td>0</td>
</tr>
<tr>
<td>5877229516333303832</td>
<td>1</td>
<td>745</td>
<td>40</td>
<td>1</td>
<td>518</td>
<td>4</td>
<td>3</td>
<td>1</td>
<td>6</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>268435987</td>
<td>37.2894</td>
<td>0.00122532</td>
<td>179.3932</td>
<td>0</td>
</tr>
<tr>
<td>5877229516333303833</td>
<td>1</td>
<td>745</td>
<td>40</td>
<td>1</td>
<td>518</td>
<td>5</td>
<td>3</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>268435986</td>
<td>37.10624</td>
<td>0.007670765</td>
<td>179.543</td>
<td>0</td>
</tr>
<tr>
<td>5877229516333303834</td>
<td>1</td>
<td>745</td>
<td>40</td>
<td>1</td>
<td>518</td>
<td>6</td>
<td>3</td>
<td>1</td>
<td>6</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>17590454611510</td>
<td>43.78992</td>
<td>0.01505366</td>
<td>187.507</td>
<td>0</td>
</tr>
<tr>
<td>5877229516333303835</td>
<td>1</td>
<td>745</td>
<td>40</td>
<td>1</td>
<td>518</td>
<td>7</td>
<td>3</td>
<td>1</td>
<td>6</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>17590454648672</td>
<td>43.60873</td>
<td>0.00538188</td>
<td>187.9388</td>
<td>0</td>
</tr>
<tr>
<td>5877229516333303836</td>
<td>1</td>
<td>745</td>
<td>40</td>
<td>1</td>
<td>518</td>
<td>8</td>
<td>3</td>
<td>1</td>
<td>6</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>448635372704330</td>
<td>-60.17922</td>
<td>0.007902309</td>
<td>496.6865</td>
<td>0</td>
</tr>
<tr>
<td>5877229516333303837</td>
<td>1</td>
<td>745</td>
<td>40</td>
<td>1</td>
<td>518</td>
<td>9</td>
<td>3</td>
<td>1</td>
<td>11</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>104668872308748</td>
<td>48.23613</td>
<td>0.1025441</td>
<td>567.665</td>
<td>0</td>
</tr>
<tr>
<td>5877229516333303838</td>
<td>1</td>
<td>745</td>
<td>40</td>
<td>1</td>
<td>518</td>
<td>10</td>
<td>3</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>483767533061264</td>
<td>48.23613</td>
<td>0.1025441</td>
<td>567.665</td>
<td>0</td>
</tr>
</tbody>
</table>
Predicates (or Conditions)

- These inequalities can be used in predicates:
  
  \[
  \begin{align*}
  &= \\
  &> \\
  &< \\
  &\geq \\
  &\leq \\
  &<> \quad (\text{“not equal”})
  \end{align*}
  \]

```
SELECT COUNT(*)
FROM PhotoObjAll
WHERE PetroMag_r < 17.7
```

It means:
Count the number of rows in PhotoObjAll where the r-band measured magnitude is brighter than 17.7.
ORDER BY

- We use “ORDER BY” to sort the output into increasing order.

```sql
SELECT Top 10 ObjID, petroMag_r
FROM PhotoObjAll
WHERE petroMag_r < 17.7
ORDER BY ObjID
```
Nullable?

- A field that is allowed to have no values is called “nullable”.
- Determined when creating the database.
- In SDSS, many unavailable fields have values “-9999”.

```sql
SELECT Top 10 ObjID, petroMag_r
FROM PhotoObjAll
WHERE petroMag_r < 17.7
AND petroMag_r <> -9999
ORDER BY ObjID
```
Aggregate ("Bag") Functions:
Group a field from multiple rows together

- Commonly used aggregate functions include:
  - \text{COUNT}(\cdot)
  - \text{MIN}(\cdot)
  - \text{MAX}(\cdot)
  - \text{AVG}(\cdot)
  - \text{STDEV}(\cdot)

- For big tables, aggregate functions may take a long time to finish.

```sql
SELECT COUNT(*) as 'Count', MIN(z) as 'Min z', MAX(z) as 'Max z', AVG(z) as 'Avg z', STDEV(z) as 'SD z'
FROM SpecObjAll
WHERE specClass = 2
```
GROUP BY

• We use “GROUP BY” to group output by column(s).
• Often used together with aggregate functions.

```
SELECT specClass, COUNT(*) as 'Count', MIN(z) as 'Min z', MAX(z) as 'Max z', AVG(z) as 'Avg z', STDEV(z) as 'SD z'
FROM SpecObjAll
WHERE z <> -9999
GROUP BY specClass
```
We use "GROUP BY" to group output by column(s).

Often used together with aggregate functions.

```
SELECT specClass, COUNT(*) as 'Count', MIN(z) as 'Min z', MAX(z) as 'Max z', AVG(z) as 'Avg z', STDEV(z) as 'SD z'
FROM SpecObjAll
WHERE z <> -9999
GROUP BY specClass
```
SQL Example:
Create Binned Redshift Histogram of Galaxies

• Suppose we want to know the redshift distribution of galaxies from the SDSS spectroscopic data.

```
DECLARE @binsize FLOAT
SET @binsize = 0.1

SELECT FLOOR(z / @binsize) * @binsize as 'Redshift',
     COUNT(*)  as 'Number of Galaxies'
FROM THUMPER.BESTDR7.dbo.SpecObjAll
WHERE specClass = 2
AND z BETWEEN 0 and 1
GROUP BY FLOOR(z / @binsize) * @binsize
ORDER BY FLOOR(z / @binsize) * @binsize
```
SQL Example:
Create Binned 2D (Redshift, Magnitude) Histogram of Galaxies

- We can select fields from multiple tables.
- We can also use the clause “JOIN” explicitly for this example.

```
DECLARE @binsize_z FLOAT
DECLARE @binsize_m FLOAT
SET @binsize_z = 0.1
SET @binsize_m = 0.2

SELECT FLOOR(s.z / @binsize_z) * @binsize_z as 'Redshift',
     FLOOR(petroMag_r / @binsize_m) * @binsize_m as 'Magnitude',
     COUNT(*) as 'Number of Galaxies'
FROM THUMPER.BESTDR7.dbo.SpecObjAll s, THUMPER.BESTDR7.dbo.PhotoObjAll p
WHERE specClass = 2
     AND s.bestObjID = p.objID
     AND s.z BETWEEN 0 and 0.2
     AND p.petroMag_r BETWEEN 16.7 and 17.7
GROUP BY FLOOR(s.z / @binsize_z) * @binsize_z,
         FLOOR(petroMag_r / @binsize_m) * @binsize_m
ORDER BY FLOOR(s.z / @binsize_z) * @binsize_z,
         FLOOR(petroMag_r / @binsize_m) * @binsize_m
```
Automated data analysis:

- Select data from DB using C# routines with SQL scripts embedded
- Perform computations
- Output results to DB, if necessary

Data Analysis using Database

1/23/2014
JHU Intersession Course - C. W. Yip

(MS SQL Server. Source: Alex Szalay)
CasJobs

- Available for public.
- Users can register and search the public SDSS data.
- All SDSS data will become public some time after the survey completes.

Exercise: Create an account in CasJobs.
Open SkyQuery

- An ambitious platform for storing and cross-matching Catalogs from many Astronomy surveys.
Hooking Up Database using R

• Here we use Microsoft Windows Operating System.
• Two main steps:
  – Set up user’s Data Source Name (DNS) in Windows.
  – Install R library for Open Database Connectivity (RODBC).
• See class demonstration.
• The R script can be downloaded from the Course Website.
Big Astronomy Databases

Large Hadron Collider
15,360 TB

SDSS (now)
Tweets in 2012
19 TB

LSST (2022)

Business Email Per Year
2,986,100 TB

Facebook Content Per Year
182,500 TB

1 Mega = 1,000,000 = 10^6
1 Giga = 10^9
1 Tera = 10^{12}
1 Peta = 10^{15}
1 Exa = 10^{18}
1 Zetta = 10^{21}

(WIRED, May 2013)