

Data Science & Analytics

HIREN DELIWALA & DR. JIAN ZHANG

A solid orange horizontal bar spanning the width of the slide at the bottom.

SQL Basics

Agenda

Introduction

DDL

DML

SELECT Statement

Structured Query Language (SQL)

What is SQL?

The ANSI standard language for the definition and manipulation of relational database.

Structured Query Language

Communicate with databases

Used to created and edit databases.

Also used to create queries, forms, and reports

Structured Query Language (SQL)

Data Definition Language (DDL)

- Commands that define a database, including creating, altering, and dropping tables and establishing constraints

Data Manipulation Language (DML)

- Commands that maintain and query a database

Data Control Language (DCL)

- Commands that control a database, including administering privileges and committing data

Schema

- The structure that contains descriptions of objects created by a user (base tables, views, constraints)

Table Basics

- A Table is an object
- Database data is stored in Tables
- Each table has a unique name
- Columns have various attributes, such as column name and data type
- Rows contain records or data for the columns

Attribute Types

Numeric Types

- Integer – integer, int, smallint, long
- Floating Point – float, real, double, precision
- Formatted – decimal (a,b), dec (l,j)

Character-String Types

- Fixed Length – char (n), character (n)
- Varying Length – varchar (n),

Bit-String Types

- Fixed Length – bit(n)
- Varying length – bit varying (n)

Date and Time Types

- Date, Time, Date time, timestamp, time with time zone, interval

Large Types

- Character – Long varchar(n), text
- Binary - blob

Table name

Attribute names

Tables in SQL

Product

PName	Price	Category	Manufacturer
Gizmo	\$19.99	Gadgets	GizmoWorks
Powergizmo	\$29.99	Gadgets	GizmoWorks
SingleTouch	\$149.99	Photography	Canon
MultiTouch	\$203.99	Household	Hitachi

Tuples or rows

Data Definition Language (DDL)

CREATE TABLE: used to create a table.

ALTER TABLE: modifies a table after it was created.

DROP TABLE: removes a table from a database.

DDL: Creating Tables

The statement to use is create table

Here is the syntax:

```
create table "tablename"  
(  
  "columnname", "datatype",  
  "columnname2", "datatype",  
  "columnname3", "datatype");
```

DDL: Creating Tables cont'd

Here is a real example:

```
create table employee  
(first varchar(15),  
last varchar(20),  
age number(3),  
address varchar(30),  
city varchar(20),  
state varchar(20));
```

DDL: Creating Tables - Steps

Use the command create table

Follow it with the correct table name

Put a parenthesis and type in the first column name

Follow it with the variable type (we will list them in a minute) then a comma

Continue the previous two steps until you have all your columns accounted for

Then put a parenthesis to close the columnname section and add a ; after it

DDL: Creating Tables - Constraints

A constraint is a rule.

We apply constraints on columns so that we can ensure good data is added into the database and the amount of NULL values is minimized

Constraints on Columns:

- NOT NULL – Attribute may not take a NULL value
- DEFAULT – Store a given default value if no value is specified
- PRIMARY KEY – Indicate which column(s) form the primary key
- FOREIGN KEY – Indicate which column(s) form a foreign key. This enforces referential integrity
- UNIQUE – Indicates which column(s) must have unique value

SQL: ALTER TABLE Statement

Use: To add or drop columns on existing tables.

ALTER TABLE statement syntax:

ALTER TABLE <table name>

ADD attribute datatype;

or

DROP COLUMN attribute;

SQL: DROP TABLE Statement

Deletes all information about the dropped relation from the database.

CASCADE: Specifies that any foreign key constraint violations that are caused by dropping the table will cause the corresponding rows of the related table to be deleted.

RESTRICT: blocks the deletion of the table if any foreign key constraint violations would be created.

DROP TABLE statement syntax:

```
DROP TABLE <table name> [ RESTRICT | CASCADE ];
```

Example:

CREATE TABLE Company

(

Name varchar(50),

Ticker varchar(5),

Location varchar (100)

);

ALTER TABLE Company (

ADD IPO_Value int

);

ALTER TABLE Company(

DROP COLUMN Location

);

DROP TABLE Company;

Company

Name	Ticker	Location
------	--------	----------

Company

Name	Ticker	Location	IPO_Value
------	--------	----------	-----------

Company

Name	Ticker	IPO_Value
------	--------	-----------

SQL: DML Commands

INSERT: adds new rows to a table.

UPDATE: modifies one or more attributes.

DELETE: deletes one or more rows from a table.

Inserting Information into Tables

To insert into tables you need only use the keyword insert. Here is the syntax:

```
insert into "tablename"  
(“first_column”, ..., “last_column”)  
values (“first_value”, ..., “last value”);
```

Inserting Information into Tables

Here is a practical example:

```
insert into employees
```

```
(first, last, age, address, city, state)
```

```
values ( 'Luke', 'Duke', 45, '2130 Boars Nest',
```

```
'Hazard Co', 'Georgia');
```

Inserting Information into Tables Steps

****All strings should be enclosed by single quotes: 'string'****

Use the keyword "insert into" followed by the tablename

Then on the next line, in parenthesis, list all the columns you are inserting values for.

Order matters – Then on the line after, type values, then in parenthesis, put the values in the same order as the columns they belong to

SQL: INSERT Statement

To insert a row into a table, it is necessary to have a value for each attribute, and order matters.

INSERT into <table name>

VALUES ('value1', 'value2', NULL);

Example: INSERT into Company

VALUES ('Apple', 'AAPL', 'Cupertino');

Company

Name	Ticker	Location
Apple	AAPL	Cupertino

SQL: UPDATE Statement

To update the content of the table:

UPDATE statement syntax:

UPDATE <table name> SET <attr> = <value>

WHERE <selection condition>;

Example: UPDATE Company SET Location = 'Baton Rouge'

WHERE Ticker = 'AAPL';

Name	Ticker	Location
Apple	AAPL	Baton Rouge

SQL: DELETE Statement

To delete rows from the table:

DELETE statement syntax:

DELETE FROM <table name>

WHERE <condition>;

Example: DELETE FROM Company

WHERE Ticker= 'AAPL';

Note: If the WHERE clause is omitted all rows of data are deleted from the table.

Deleting Records Examples

Employee Table – employee id, firstname, lastname, address, Dob

ex) delete from employees;

deletes all records from that table

ex) delete from employee

where lastname='May';

deletes all records for people whose last name is May

ex) delete from employee

where firstname='Mike' or firstname='Eric';

deletes all records for anyone whose first name is Mike or Eric

Congratulations You Have Learned Basics of SQL

EXCEPT SELECT

SQL SELECT

SQL SELECT

INTRODUCTION

Basic Query Structure

A typical SQL query has the form:

```
select A1, A2, ..., An  
from r1, r2, ..., rm  
where P
```

- A_i represents an attribute
- R_i represents a relation
- P is a predicate.

SQL: SELECT Statement

A basic SELECT statement includes 3 clauses

SELECT <attribute name> FROM <tables> WHERE <condition>

SELECT

- Specifies the attributes that are part of the resulting relation

FROM

- Specifies the tables that serve as the input to the statement

WHERE

- Specifies the selection condition, including the join condition.

Note: that you don't need to use WHERE

SQL Query

Basic form: (plus many many more bells and whistles)

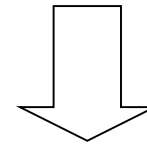
```
SELECT <attributes>  
FROM   <one or more relations>  
WHERE  <conditions>
```

Notation

Input Schema

Product(PName, Price, Category, Manufacturer)

```
SELECT PName, Price, Manufacturer
FROM   Product
WHERE  Price > 100
```



Answer(PName, Price, Manufacturer)

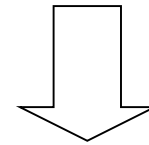
Output Schema

Simple SQL Query

Product

PName	Price	Category	Manufacturer
Gizmo	\$19.99	Gadgets	GizmoWorks
Powergizmo	\$29.99	Gadgets	GizmoWorks
SingleTouch	\$149.99	Photography	Canon
MultiTouch	\$203.99	Household	Hitachi

```
SELECT *  
FROM Product  
WHERE category='Gadgets'
```



PName	Price	Category	Manufacturer
Gizmo	\$19.99	Gadgets	GizmoWorks
Powergizmo	\$29.99	Gadgets	GizmoWorks

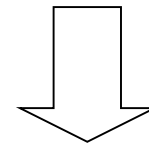
“selection”

Simple SQL Query

Product

PName	Price	Category	Manufacturer
Gizmo	\$19.99	Gadgets	GizmoWorks
Powergizmo	\$29.99	Gadgets	GizmoWorks
SingleTouch	\$149.99	Photography	Canon
MultiTouch	\$203.99	Household	Hitachi

```
SELECT PName, Price, Manufacturer
FROM   Product
WHERE  Price > 100
```



“selection” and
“projection”

PName	Price	Manufacturer
SingleTouch	\$149.99	Canon
MultiTouch	\$203.99	Hitachi

Example: Person

Name	Age	Weight
Harry	34	80
Sally	28	64
George	29	70
Helena	54	54
Peter	34	80

2) SELECT weight
FROM person
WHERE age > 30;

Weight
80
54
80

1) SELECT *
FROM person
WHERE age > 30;

Name	Age	Weight
Harry	34	80
Helena	54	54
Peter	34	80

3) SELECT **distinct** weight
FROM person
WHERE age > 30;

Weight
80
54

Details

Case insensitive:

- Same: SELECT Select select
- Same: Product product
- Different: 'Seattle' 'seattle'

Constants:

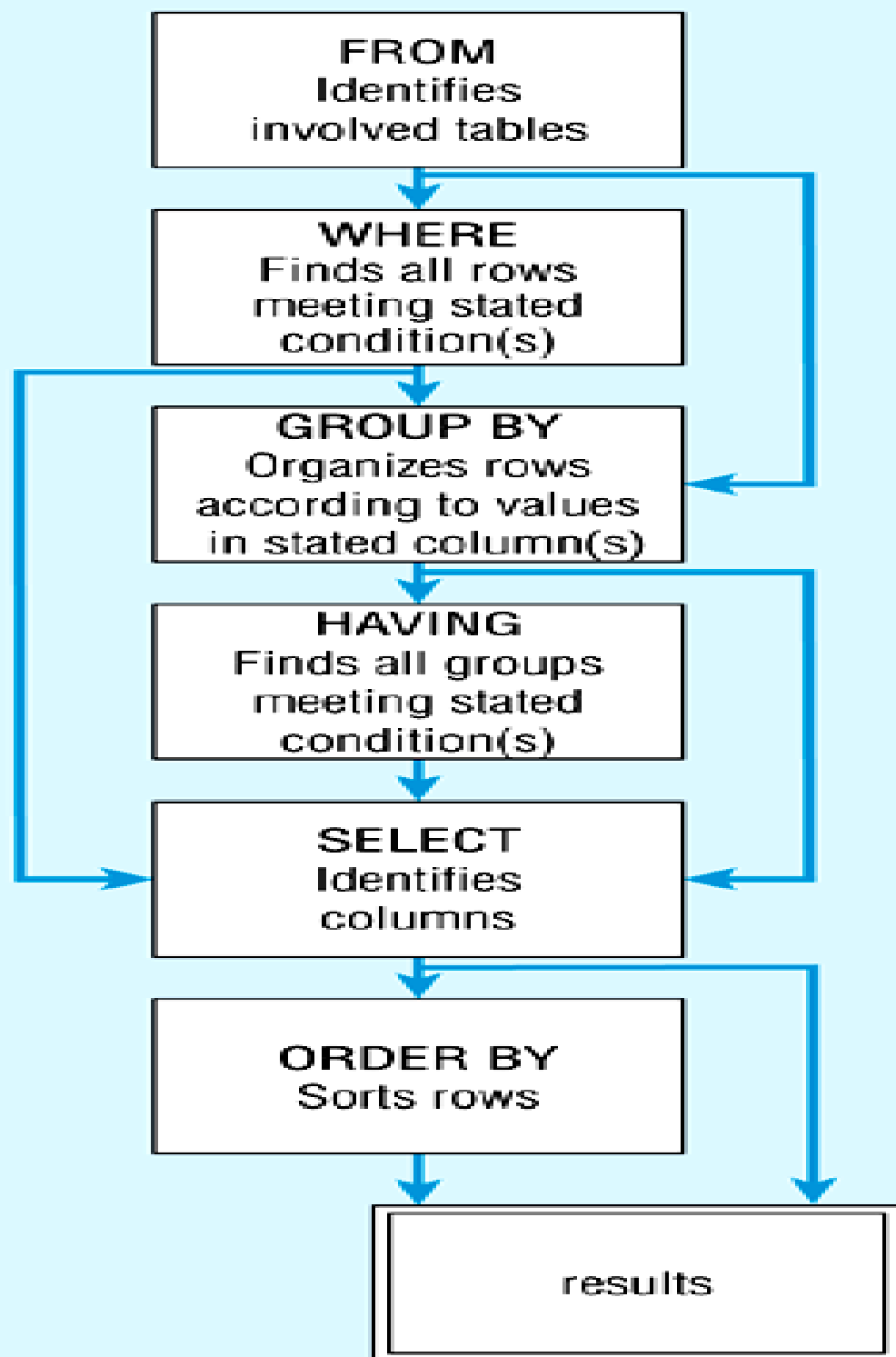
- 'abc' - yes
- "abc" - no

SELECT Statement

Used for queries on single or multiple tables

Clauses of the SELECT statement:

- **SELECT**
 - List the columns (and expressions) that should be returned from the query
- **FROM**
 - Indicate the table(s) or view(s) from which data will be obtained
- **WHERE**
 - Indicate the conditions under which a row will be included in the result
- **GROUP BY**
 - Indicate columns to group the results
- **HAVING**
 - Indicate the conditions under which a group will be included
- **ORDER BY**
 - Sorts the result according to specified columns

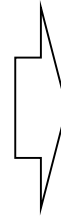


SQL statement processing order

SQL Concepts

Eliminating Duplicates

```
SELECT DISTINCT category  
FROM Product
```



Category
Gadgets
Photography
Household

Compare to:

```
SELECT category  
FROM Product
```



Category
Gadgets
Gadgets
Photography
Household

Ordering the Results

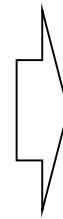
```
SELECT pname, price, manufacturer  
FROM Product  
WHERE category='gizmo' AND price > 50  
ORDER BY price, pname
```

Ties are broken by the second attribute on the ORDER BY list, etc.

Ordering is ascending, unless you specify the DESC keyword.

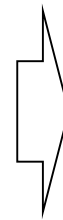
PName	Price	Category	Manufacturer
Gizmo	\$19.99	Gadgets	GizmoWorks
Powergizmo	\$29.99	Gadgets	GizmoWorks
SingleTouch	\$149.99	Photography	Canon
MultiTouch	\$203.99	Household	Hitachi

```
SELECT DISTINCT category
FROM Product
ORDER BY category
```



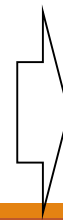
?

```
SELECT Category
FROM Product
ORDER BY PName
```



?

```
SELECT DISTINCT category
FROM Product
ORDER BY PName
```



?

SELECT Example using Alias

Alias is an alternative column or table name

- `SELECT CUST.CUSTOMER AS NAME, CUST.CUSTOMER_ADDRESS`
- `FROM CUSTOMER_V CUST`
- `WHERE NAME = 'Home Furnishings';`

SELECT Example – Boolean Operators

AND, OR, and NOT Operators for customizing conditions in WHERE clause

- SELECT PRODUCT_DESCRIPTION, PRODUCT_FINISH, STANDARD_PRICE
- FROM PRODUCT_V
- WHERE (PRODUCT_DESCRIPTION LIKE '%Desk'
- OR PRODUCT_DESCRIPTION LIKE '%Table')
- AND UNIT_PRICE > 300;

Note: the LIKE operator allows you to compare strings using wildcards. For example, the % wildcard in '%Desk' indicates that all strings that have any number of characters preceding the word “Desk” will be allowed

SQL: Aggregate Functions

Are used to provide summarization information for SQL statements, which return a single value.

COUNT(attr)

SUM(attr)

MAX(attr)

MIN(attr)

AVG(attr)

Note: when using aggregate functions, NULL values are not considered, except in COUNT(*) .

SELECT Example Using a Function

Using the COUNT aggregate function to find totals

Aggregate functions: SUM(), MIN(), MAX(), AVG(), COUNT()

- `SELECT COUNT(*) FROM ORDER_LINE_V`
- `WHERE ORDER_ID = 1004;`

SQL: Aggregate Functions

FoodCart

date	food	sold
02/25/08	pizza	349
02/26/08	hotdog	500
02/26/08	pizza	70

COUNT(attr) -> return # of rows that are not null

- Ex: COUNT(distinct food) from FoodCart; -> 2

SUM(attr) -> return the sum of values in the attr

- Ex: SUM(sold) from FoodCart; -> 919

MAX(attr) -> return the highest value from the attr

- Ex: MAX(sold) from FoodCart; -> 500

SQL: Aggregate Functions

FoodCart

date	food	sold
02/25/08	pizza	349
02/26/08	hotdog	500
02/26/08	pizza	70

MIN(attr) -> return the lowest value from the attr

- Ex: MIN(sold) from FoodCart; -> 70

AVG(attr) -> return the average value from the attr

- Ex: AVG(sold) from FoodCart; -> 306.33

Note: value is rounded to the precision of the datatype

How Do I Get Data From Multiple Tables?

JOINS

A solid orange horizontal bar spanning the width of the slide at the bottom.

SQL Statements, Operations, Clauses

Select Operations:

- Join
- Left Join
- Right Join
- Like

Select Clauses:

- Order By
- Group By
- Having

Keys and Foreign Keys

Company



<u>CName</u>	StockPrice	Country
GizmoWorks	25	USA
Canon	65	Japan
Hitachi	15	Japan

Product

<u>PName</u>	Price	Category	Manufacturer
Gizmo	\$19.99	Gadgets	GizmoWorks
Powergizmo	\$29.99	Gadgets	GizmoWorks
SingleTouch	\$149.99	Photography	Canon
MultiTouch	\$203.99	Household	Hitachi



Foreign
key

Joins

Product (pname, price, category, manufacturer)

Company (cname, stockPrice, country)

Find all products under \$200 manufactured in Japan;
return their names and prices.

```
SELECT PName, Price  
FROM Product, Company  
WHERE Manufacturer=CName AND Country='Japan'  
AND Price <= 200
```



Join
between Product
and Company

Joins

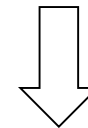
Product

PName	Price	Category	Manufacturer
Gizmo	\$19.99	Gadgets	GizmoWorks
Powergizmo	\$29.99	Gadgets	GizmoWorks
SingleTouch	\$149.99	Photography	Canon
MultiTouch	\$203.99	Household	Hitachi

Company

Cname	StockPrice	Country
GizmoWorks	25	USA
Canon	65	Japan
Hitachi	15	Japan

```
SELECT PName, Price
FROM Product, Company
WHERE Manufacturer=CName AND Country='Japan'
AND Price <= 200
```



PName	Price
SingleTouch	\$149.99

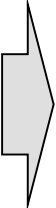
Tuple Variables

Person(pname, address, worksfor)

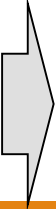
Company(cname, address)

```
SELECT DISTINCT pname, address
FROM Person, Company
WHERE worksfor = cname
```

Which
address ?



```
SELECT DISTINCT Person.pname, Company.address
FROM Person, Company
WHERE Person.worksfor = Company.cname
```



```
SELECT DISTINCT x.pname, y.address
FROM Person AS x, Company AS y
WHERE x.worksfor = y.cname
```

SQL Clauses

- Order By
- Group By
- Having

ORDER BY Example

Sort the results first by STATE, and within a state by CUSTOMER_NAME

- SELECT CUSTOMER_NAME, CITY, STATE
- FROM CUSTOMER_V
- WHERE STATE IN ('FL', 'TX', 'CA', 'HI')
- ORDER BY STATE, CUSTOMER_NAME;

Note: the IN operator in this example allows you to include rows whose STATE value is either FL, TX, CA, or HI. It is more efficient than separate OR conditions

GROUP BY Clause

- SELECT STATE, COUNT(STATE)
- FROM CUSTOMER_V
- GROUP BY STATE;
- Note: you can use single-value fields with aggregate functions if they are included in the GROUP BY clause

Using the HAVING Clause

For use with GROUP BY

- SELECT STATE, COUNT(STATE)
 - FROM CUSTOMER_V
 - GROUP BY STATE
 - HAVING COUNT(STATE) > 1;
-
- Like a WHERE clause, but it operates on groups (categories), not on individual rows. Here, only those groups with total numbers greater than 1 will be included in final result