

SQL for Single Table Queries

PLAN for 11/15

- SQL Syntax – SINGLE Tables
 - Types of Returned Results: (SELECT...)
 - Functions: Count, Max, Min, Avg, Sum
 - Distinct
 - Constraining the Returned Results (WHERE...)
 - Comparison Operators
 - Boolean operators
 - Ranges and “BETWEEN”
 - Wildcards and “Like”
 - Sets and “IN”
 - Organizing the Returned Results (WHERE...)
 - ORDER BY
 - GROUP BY
 - HAVING
- Timing
 - Look over results (groups, 5-10 minutes)
 - Identify examples of syntax ...
 - Discuss sets of SQL Syntax (class/groups, rest of class)
 - Discuss element of syntax
 - Examples from book, from class prep
 - Closing Comments about
 - Questions
 - Next class prep

SQL - Types of Returned Results

Using Functions

- Description: Functions such as COUNT, Min, MAX, SUM, AVG of specified columns in the column list of a select command may be used to specify that the resulting answer table is to contain aggregated data instead of row level data.
 - SUM, AVG: Only for numeric data types
 - MIN, MAX, COUNT: For all data types

- Examples:

How many products are on the product table?

```
SELECT COUNT(PRODUCT_DESCRIPTION)
FROM PRODUCT_TABLE;
```

What is alphabetically, the first product name on the table?

```
SELECT MIN(PRODUCT_NAME)
FROM PRODUCT_TABLE;
```

- Example from Class Prep Activity:

SQL - Types of Returned Results

Distinct

- Sometimes when returning rows that do not include the primary key, duplicate rows will be returned. If, however, we add the keyword “DISTINCT”, then only one occurrence of a value will be returned.

- Examples

What order ids exist?

```
SELECT ORDER_ID  
FROM ORDER_LINE_T;
```

What distinct order ids exist?

```
SELECT DISTINCT ORDER_ID  
FROM ORDER_LINE_T;
```

- EXAMPLES FROM CLASS PREP?

SQL - Constraining Returned Results

Comparison Operators

- Description: Comparison operators can be used to constrain the WHERE clause in a select statement
 - Operators include: =, >, >=, <, <=, <>, !=

- Examples:

What orders came in after 10/24/98?

```
SELECT ORDER_ID, ORDER_DATE
       FROM ORDER_T
       WHERE ORDER_DATE > '24-Oct-98';
```

What are the products and finishes that are not cherry?

```
SELECT PRODUCT_NAME, PRODUCT_FINISH
       FROM PRODUCT_T
       WHERE PRODUCT_FINISH != "Cherry";
```

- Example from classprep...

SQL - Constraining Returned Results

Boolean Operators

- Boolean operators (and order of operations) can be used to modify the where clause further...
 - AND: Joins two or more conditions and returns results only when all conditions are true
 - OR: Joins two or more conditions and returns results when any conditions are true
 - NOT: Negates an expression

- Example:

What are names, finish and unit price for all desks and only tables that cost more than 300?

```
SELECT PRODUCT_NAME, PRODUCT_FINISH, UNIT_PRICE
FROM PRODUCT_T
WHERE PRODUCT_NAME LIKE '%Desk' or PRODUCT_NAME
LIKE '%Table' and UNIT_PRICE > 300;
```

What are names, finish and unit price for desks and tables that cost more than 300?

```
SELECT PRODUCT_NAME, PRODUCT_FINISH, UNIT_PRICE
FROM PRODUCT_T
WHERE (PRODUCT_NAME LIKE '%Desk' or
LIKE '%Table') and UNIT_PRICE > 300;
```

- Example from classroom preparation?

SQL - Constraining Returned Results Using Wildcards

- Description: Wildcards may be used in the WHERE clause where an exact match is not possible. The keyword “LIKE” is paired with wildcard characters and usually a string containing the characters known to be desired matches.
 - “_” or “*” represents any single character
 - “%” represents any collection of characters

- Examples -

What are “desk” products?

```
SELECT PRODUCT_NAME  
FROM PRODUCT_TABLE  
WHERE PRODUCT_NAME LIKE “%Desk”;
```

Please list all products that have a single digit number of drawers.

```
SELECT PRODUCT_NAME  
FROM PRODUCT_TABLE  
WHERE PRODUCT_NAME LIKE “_ -drawer”;
```

- Example from Class Prep Activity:

SQL - Constraining Returned Results

Ranges

- Ranges of possible values can be indicated in WHERE clauses via two mechanisms
 - using the Comparison operators < and >
 - Using “between” or “not between”

- Examples

What are products and prices for products between 200 and 300?

```
SELECT PRODUCT_NAME, UNIT_PRICE  
FROM PRODUCT_T  
WHERE UNIT_PRICE > 199 AND UNIT PRICE < 301
```

What are products and prices for products between 200 and 300?

```
SELECT PRODUCT_NAME, UNIT_PRICE  
FROM PRODUCT_T  
WHERE UNIT_PRICE BETWEEN 199 AND 301
```

- Examples from Class Prep

SQL - Constraining Returned Results

“In” and “Not in” Lists

- A where clause can be modified to test whether an attribute's value is in a given set via the “In” and “Not in” operators.
 - IN: provides values to include
 - NOT IN: provides values to exclude.

- Examples:

Who are customers in warm states (i.e., florida, texas...)?

```
SELECT CUSTOMER_NAME, CITY, STATE
FROM CUSTOMER_T
WHERE STATE IN ('FL', 'TX', 'CA', 'HI');
```

- Examples from class prep activity?

SQL – Organizing Results

Sorting via “Order by”

- The results of an SQL query can be sorted into ascending or descending order via the “order by” element. “Order by” is followed by the names of the attributes for which the SQL results are to be ordered.

- Examples:

Please list all customers in warm states (i.e., florida, texas...), ordering the list by state.

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

- Examples from class prep?

SQL – Organizing Results

Categorizing via “Group By”

- GROUP BY is particularly useful when paired with aggregate functions, such as SUM or COUNT. GROUP BY divides a table into subsets (by groups); then an aggregate function (count, sum, min, max, avg) can be used to provide summary information for that group. It is also possible to nest groups.

- Example:

How many customers are in each state?

```
SELECT STATE, COUNT(STATE)
FROM CUSTOMER_T
GROUP BY STATE;
```

How many customers are in each city?

```
SELECT STATE, CITY, COUNT(CITY)
FROM CUSTOMER_T
GROUP BY STATE, CITY;
```

- Examples from class prep?

SQL – Organizing Results

Restricting categories via HAVING

- The HAVING clause acts like a where clause, but it identifies groups that meet a criterion, rather than rows. Therefore, one usually sees a HAVING clause following a group clause.

- Example:

How many customers are in states with more than 1 customer?

```
SELECT STATE, COUNT(STATE)
FROM CUSTOMER_T
GROUP BY STATE
HAVING COUNT(STATE)>1;
```

- Examples from class prep?