Using String Patterns, Rangers, and Sets

* Retrieval of data
  + **Select \* from Book** (This will retrieve everything from book)
  + **Select book\_id, title from Book** (This will retrieve the book id and title from book only)
  + **Select book\_id, title from Book where book\_id = ‘b1’** (This is more specific to only retrieve the book id and title of b1.)
* String patterns
  + Like function
  + **WHERE firstname Like R%** (Search for a firstname that starts with R)
  + % Sign is a wild card character used to substitute characters. (Replace it with var you don’t know)
  + Graphical user interface, text

    Description automatically generatedExamples
    - **Select firstname from Author WHERE firstname like ‘R%’**
    - Image on the right should be the results.
* Retrieving rows – using a Range
  + Example
    - **Select title, pages from Book WHERE pages >=290 AND pages <=300**
    - Not the ‘and’ functions as a range.,
  + Another Example
    - **Select title, pages from Book WHERE pages between 290 and 300**
  + To retrieve a range with a string
    - **Graphical user interface, text

      Description automatically generated with medium confidenceSelect firstname, lastname, country from Author**

Sorting Results Sets

* ORDER BY clause
  + General Retrieval
    - **Select title from Book**
  + Ordered Retrieval
    - **Select title from Book ORDER BY title** (Defaults to ascending order)
    - **Select title from Book ORDER BY title DESC** (Now in Descending order)
    - ***Select title from Book ORDER BY title DESC nulls last \*\*****REMOVES NULLS DATA POINTS*
    - **SELECT \* FROM EMPLOYEES ORDER BY DEP\_ID DESC, L\_NAME DESC;**
  + Specifying Column Sequence Number
    - **Table

      Description automatically generated with medium confidenceSelect title, pages from Book ORDER BY 2** (The number 2 indicate the Column in this case Column 2 which is ‘pages)

Grouping Results Sets

* Eliminating Duplicates – DISTINCT clause
  + Select country from Author ORDER BY 1 (Will Yield some duplicates if there are 2 or more authors coming from the same country)
  + Distinct clause
    - Example: **select distinct(country) from Author**
  + GROUP BY clause
    - Example: **select country, count(country) from Author GROUP BY country**

**Graphical user interface, text, application, chat or text message

Description automatically generated(This yields the amount of authors from that country)**

* + - Table

      Description automatically generatedNaming the column: **select country, count(country) from Author as Count from Author group by country**
    - **SELECT DEP\_ID, AVG(SALARY) as AvgSal, count(DEP\_ID) as NumOfEmply FROM EMPLOYEES GROUP BY DEP\_ID;**
  + Graphical user interface, text, application

    Description automatically generatedHAVING clause
    - Example: **select country, count(country)**

**as Count from Author**

**group by country**

**having count(country) > 4**

(This sets a condition to group countries with more than 4 author)

* + Combining Conditions
    - For each department Id retrieve the number of employees in the department.
    - For each department retrieve the number of employees in the department, and the average employee salary in the department.
    - Ordered by Average Salary
    - Limit the result to departments with fewer than 4 employees.

**SELECT DEP\_ID, ##Selecting DEPARTMENT ID**

**AVG(SALARY) as "Average Salary", ##Getting a column for Average Salary and naming it.**

**count(DEP\_ID) as "Number Of Employees" ##counting number of entities with same dep id.**

**FROM EMPLOYEES ##Retrieving info from employees**

**GROUP BY DEP\_ID ##Group by Dep ID**

**HAVING count(\*) < 4 ##Limiting to fewer than 4 employees**

**ORDER BY AVG(SALARY); ##Order by avg Salary in ascending order.**

Functions, Multiple Tables, and Sub-queries

* Built-In Functions
  + Aggregate
    - Input: Collection of Values
    - Output: Single Values
    - Examples: SUM(), MIN(), MAX(), AVG()
      * **SUM(COLUMN\_NAME)**
        + **Select SUM(COST) from PETRESCUE;** –RETURNS SUM of an entire column
        + **Select SUM(COST) as SUM\_OF\_COST from PETRESCUE;** \*\*This names the sum(cost).
      * MIN(), MAX()
        + **Select MAX(QUANTITY) from PETRESCUE;**
        + **Select MIN(ID) from PETRESCUE where ANIMAL = ‘Dog’;**
      * AVG()
        + **Select AVG(COST) from PETRESCUE;**
        + **Select AVG(COST / QUANTITY) from PETRESCUE where ANIMAL = ‘Dog’; \*\*Mathematical operations within the avg function.**
    - SCALAR and String Functions
      * ROUND()
        + **Select ROUND(COST) from PETRESUE;** (Round up or down costs)
      * LENGTH() \*\*For Char or Varchar
        + **Select LENGTH(ANIMAL) from PETRESCUE;** (Return the string length of the data.)
      * UCASE
        + **Select UCASE(ANIMAL) from PETRESCUE;**
      * LCASE
        + **Select LCASE(ANIMAL) from PETRESCUE;**
        + **Select \* from PETRESCUE where LCASE(ANIMAL) = ‘cat’**
        + **Select DISTINCT(UCASE(ANIMAL)) from PETRESCUE;**
    - Data, Time Functions
      * DATE: YYYMMDD
      * TIME: HHMMSS
      * TIMESTAMP: YYYXXDDHHMMSSZZZZZZ
      * Functions: YEAR(), MONTH(), DAY(), DAYOFMONTH(), DAYOFWEEK(), DAYOFYEAR(), WEEK(), HOUR(), MINUTE(), SECOND()
      * Examples:
        + **select DAY(RESCUEDATE) from PETRESCUE where ANIMAL = ‘cat’;** (This returns the day from the rescuedata data member only.)
        + **select COUNT(\*) from PETRESCUE where MONTH(RESCUEDATE) = ‘05’;** (This returns a number of rescues during the month of May)
      * Date or Time Arithmetic
        + **Select (RESCUEDATE + 3 DAYS) from PETRESCUE;** (Returns dates 3 days after the date listed in rescuedate)
      * CURRENT\_DATE, CURRENT\_TIME
        + **Select (CURRENT\_DATE – RESCUEDATE) from PETRESCUE;** (Returns results as YMMDD, shows how many days have passed since each rescuedate till now)
  + **Code from exercise**

**select SUM(COST)as SUM\_OF\_ALL\_ANIMALS,**

**MAX(QUANTITY) as MAX\_AMOUNT\_OF\_ANIMALS,**

**AVG(COST) as AVG\_COST\_PER\_ANIMAL**

**from PETRESCUE;**

**select AVG(COST/QUANTITY) as AVG\_COST\_PER\_DOG from PETRESCUE where ANIMAL = 'Dog';**

**select ROUND(COST) as ROUNDED\_COST,**

**LENGTH(ANIMAL) AS LENGTH\_OF\_ANIMAL\_NAME,**

**UCASE(ANIMAL) AS UPPPER\_CASE\_NAME**

**from PETRESCUE;**

**select DISTINCT(UCASE(ANIMAL)) AS DISTINCT\_ANIMALS from PETRESCUE;**

**select \* from PETRESCUE where LCASE(ANIMAL) = 'cat';**

**select DAY(RESCUEDATE) as DAY\_OF\_RESCUE from PETRESCUE where ANIMAL = 'Cat';**

**select count(\*) as NUM\_RESCUE\_MAY from PETRESCUE where MONTH(RESCUEDATE) = 05;**

**select count(\*) as NUM\_RESCUE\_DAY14 from PETRESCUE where DAY(RESCUEDATE) = 14;**

**select RESCUEDATE,(RESCUEDATE + 3 DAYS) as DATE\_TO\_VISIT\_VET from PETRESCUE;**

**select (CURRENT\_DATE - RESCUEDATE) as DATE\_RESCUE\_FROM\_TODAY from PETRESCUE;**

* + Column Function
  + Sub-queries and Nested Selects (Column Expressions)
    - Sub-queries: A query inside another query
    - Typical Nested Syntax

SELECT column\_name [, column\_name ]

FROM table1 [, table2 ]

WHERE column\_name OPERATOR

(SELECT column\_name [, column\_name ]

FROM table1 [, table2 ]

* + - WHERE condition);
      * Example: **select COLUMN1 from TABLE where COLUMN@ = (select MAX(COLUMN2) from TABLE)**
      * Sub-queries can be used to solve aggregate functions
        + IE: AVG() cannot be used in the WHERE clause
        + **Select \* from employees where salary > AVG(salary);**

The statement above produces an error. To fix below.

* + - * + **Select EMP\_ID, F\_NAME, L\_NAME, SALARY**

**from employees**

**where SALARY < (select AVG(SALARY) from employees);**

* FROM clause (Derived Tables or table expression)
  + **Select \* from**

**(select EMP\_ID, F\_NAME, L\_NAME, DEP\_ID**

**From employees) AS EMP4ALL;**

* + Execute a Table Expression for the EMPLOYEES table that excludes columns with sensitive employee data (IS NO SSN, BDATE, SEX, ADDRESS, SALARY.
  + **select \* from (select EMP\_ID, F\_NAME, L\_NAME, JOB\_ID, MANAGER\_ID, DEP\_ID from EMPLOYEES);**

**RED DOESN’T WORK FIX IS BLUE**

* + select \* from EMPLOYEES where SALARY < AVG(SALARY); \*\*WONT WORK
  + select \* from EMPLOYEES where SALARY < (select AVG(SALARY) from EMPLOYEES); \*\* THIS IS THE FIX
  + select EMP\_ID, SALARY, MAX(SALARY) as MAX\_SALARY from EMPLOYEES;
  + select EMP\_ID, SALARY, (select max(SALARY) from EMPLOYEES) as MAX\_SALARY from EMPLOYEES;

Working with multiple tables

* Ways to access multiple tables
  + Sub-queries
    - **Select \* from employees**

**Where DEP\_IN IN (select DEPT\_ID\_DEP from departments where LOC\_ID = ‘L0002’);**

* + To retrieve the department ID and name for employees who earn more than 70,000
    - **Select DEP\_ID\_DEP, DEP\_NAME from DEPARTMENTS**

**Where DEPT\_ID\_DEP in (select DEP\_ID from EMPLOYEES where SALARY > 70000);**

* + Retrieve only the list of employees whose JOB\_TITLE is Jr.Designer
    - **select \* from employees where JOB\_ID IN (select JOB\_IDENT from jobs where JOB\_TITLE= 'Jr. Designer');**
  + Implicit JOIN
    - **Select \* from employees, departments;** (Full join or Cartesian join)
    - Use additional operands to limit the results set
    - **Select \* from employees, departments**

**Where employees.DEP\_ID =**

**departments.DEPT\_ID\_DEP;**

* + - Shorten the name
    - **Select \* from employees E, departments D**

**Where E.DEP\_ID = D.DEPT\_ID\_DEP;**

* + Graphical user interface, text, application

    Description automatically generatedJOIN operators (INNER JOIN, OUTER JOIN, etc.)

EXERCISE 1: ACCESSING MULTIPLE TABLES WITH SUB\_QUERIES

* + --Retrieve only the employees records that correspoond to jobs in the JOBS table.
  + **select \* from employees where JOB\_ID IN (select JOB\_IDENT from jobs);**
  + --Retireve only hthe list of employees whose JOB\_TITLE is Jr. Designer.
  + **select \* from employees where JOB\_ID IN (select JOB\_IDENT from jobs where JOB\_TITLE= 'Jr. Designer');**
  + --Retrieve JOB information and list of employees who earn more than $70,000
  + **select \* from JOBS where JOB\_IDENT IN(select Job\_ID from employees where SALARY > 70000);**
  + --Retrieve JOB information and list of employees whose birth year is after 1976.
  + **select \* from JOBS where JOB\_IDENT IN(select Job\_ID from employees where YEAR(B\_DATE) > 1976);**
  + --Retrieve JOB information and list of female employees whose birth year is after 1976.
  + **select \* from JOBS where JOB\_IDENT IN(select Job\_ID from employees where YEAR(B\_DATE) > 1976 and SEX = 'F');**

EXERCISE 2: ACCESSING MULTIPLE TABLES WITH IMPLICIT JOINS

Perform an implicit join between Employees and JOBS tables

* + **select \* from employees ,jobs;**
  + --Retrieve only the Employees records that correspond to jobs in the jobs table.
  + **select \* from employees, jobs where employees.JOB\_ID = jobs.JOB\_IDENT;**
  + --Redo the previous query, using shorter aliases for table names.
  + **select \* from employees E, jobs J where E.JOB\_ID = J.JOB\_IDENT;**
  + --Redo the previous query, but retrieve only the employee ID, Employee Name and Job Title.
  + **select EMP\_ID, F\_NAME, L\_NAME, JOB\_TITLE from employees E, jobs J where E.JOB\_ID = J.JOB\_IDENT;**
  + --Redo the previous query but specify the fully qualified column names with aliases in the select clause.
  + **select E.EMP\_ID, E.F\_NAME, E.L\_NAME, J.JOB\_TITLE from employees E, jobs J where E.JOB\_ID = J.JOB\_IDENT;**