*Please complete the Zoom poll as we wait for others to join.*
Agenda

- Background
- Query Components
- Joins and Unions
- Conditions
- Common Functions
- Resources
Background

- Differences between a database, data warehouse, and data lake
  - Database - place that stores a collection of data (ex: Banner db)
  - Data Warehouse - data collected from various sources and is optimized for reporting (ex: data from Banner, iModules, and AlumnIQ)
  - Data Lake - location to store large amounts of raw data (usually unorganized) from various sources
Background

- Differences between a database, data warehouse, and data lake
- Some terminology
  - Column - specific piece of data, AKA a field
  - Row - combination of columns, AKA a record
  - Table - collection of related rows/data
  - Result Set - results of a SQL query
  - Primary Key - 1+ columns used as unique identifier for each row in a table
  - Foreign Key - 1+ columns used to identify a record in another table
Background

- Differences between a database, data warehouse, and data lake
- Some terminology
- What is SQL?
  - The programming language that allows us to work with data
  - There are multiple versions out there: PL/SQL, T-SQL, PostgreSQL, MySQL, etc…
  - Is it Sequel? S-Q-L? Does it stand for Structured Query Language?
Query Components (AKA “clauses”)

- SELECT
- FROM
- WHERE
- GROUP BY
- HAVING
- ORDER BY
Query Components

- **SELECT**
  - 1st clause of a SELECT query statement
  - Determines which columns to include in the results
  - Processed last
  - Aliases can be used

- **FROM**
  - What tables or views to pull data “from”
  - Subqueries too!
  - Aliases can be used

- **WHERE**
  - Filters the data based on conditions
Query Components

- **SELECT**
- **FROM**
- **WHERE**
- **GROUP BY**
  - Grouping rows by a common value
- **HAVING**
  - Filters the group data
- **ORDER BY**
  - Sorts the result set
  - Sort by ascending (asc), descending (desc), numeric placeholders, or expressions
Examples

- Simple SELECT queries in SQL Developer
Boredom Check

Bored.
Joins and Unions

- **INNER JOIN**

  ```sql
  SELECT <fields>
  FROM TableA A
  INNER JOIN TableB B
  ON A.key = B.key;
  ```

- **FULL JOIN**

  ```sql
  SELECT <fields>
  FROM TableA A FULL JOIN TableB B
  ON A.key = B.key;
  ```

  --Add a WHERE clause with null

  ```sql
  WHERE A.key IS NULL OR B.key IS NULL
  ```
**Joins and Unions**

- **LEFT JOIN**

  ```sql
  SELECT <fields>
  FROM TableA A LEFT JOIN TableB B
  ON A.key = B.key;
  ```

- **RIGHT JOIN**

  ```sql
  SELECT <fields>
  FROM TableA A FULL RIGHT JOIN TableB B
  ON A.key = B.key;
  ```

  --Add a WHERE clause with null
  ```sql
  WHERE B.key IS NULL
  ```

  --Add a WHERE clause with null
  ```sql
  WHERE A.key IS NULL
  ```
Joins and Unions

- UNION … removes duplicates

```sql
SELECT
    'Student' as Category,
    stuName as Name,
    stuEmail as Email
FROM stuTable
UNION
SELECT
    'Employee' as Category,
    empName as Name,
    empEmail as Email
FROM empTable;
```

<table>
<thead>
<tr>
<th>Category</th>
<th>Name</th>
<th>Email</th>
</tr>
</thead>
<tbody>
<tr>
<td>Student</td>
<td>Jessica</td>
<td><a href="mailto:jes123@lehigh.edu">jes123@lehigh.edu</a></td>
</tr>
<tr>
<td>Student</td>
<td>Trent</td>
<td><a href="mailto:trw522@lehigh.edu">trw522@lehigh.edu</a></td>
</tr>
<tr>
<td>Employee</td>
<td>John</td>
<td><a href="mailto:jkz410@lehigh.edu">jkz410@lehigh.edu</a></td>
</tr>
<tr>
<td>Employee</td>
<td>Megan</td>
<td><a href="mailto:moq107@lehigh.edu">moq107@lehigh.edu</a></td>
</tr>
<tr>
<td>Employee</td>
<td>Jaida</td>
<td><a href="mailto:jlr219@lehigh.edu">jlr219@lehigh.edu</a></td>
</tr>
</tbody>
</table>
Joins and Unions

- UNION ALL … keeps duplicates

```
SELECT
    stuName as Name,
    stuEmail as Email
FROM
    stuTable
UNION ALL
SELECT
    empName as Name,
    empEmail as Email
FROM
    empTable;
```

<table>
<thead>
<tr>
<th>Name</th>
<th>Email</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jessica</td>
<td><a href="mailto:jes123@lehigh.edu">jes123@lehigh.edu</a></td>
</tr>
<tr>
<td>Trent</td>
<td><a href="mailto:trw522@lehigh.edu">trw522@lehigh.edu</a></td>
</tr>
<tr>
<td>John</td>
<td><a href="mailto:jkz410@lehigh.edu">jkz410@lehigh.edu</a></td>
</tr>
<tr>
<td>Megan</td>
<td><a href="mailto:moq107@lehigh.edu">moq107@lehigh.edu</a></td>
</tr>
<tr>
<td>Jaida</td>
<td><a href="mailto:jlr219@lehigh.edu">jlr219@lehigh.edu</a></td>
</tr>
<tr>
<td>Trent</td>
<td><a href="mailto:trw522@lehigh.edu">trw522@lehigh.edu</a></td>
</tr>
<tr>
<td>Jaida</td>
<td><a href="mailto:jlr219@lehigh.edu">jlr219@lehigh.edu</a></td>
</tr>
</tbody>
</table>
Conditions

- Basic operators and conditions
  - Pay attention to the AND/OR combinations. Apply parentheses.
  - = (equal)  ! (not equal)  <> (not equal)
  - < (less than)  > (greater than)  LIKE (wildcard, % or _)
  - + (add)  - (subtract)  IN (list of values)
  - * (multiply)  / (divide)  BETWEEN (btw values)
  - NOT is used to negate a condition

- NULL means the value does not exist or is not known yet, for a given field
  - An expression can “be” null, but it can not = null
  - Two nulls are NEVER equal to each other
  - Expressions: IS NULL and IS NOT NULL
Examples

● JOIN query examples in SQL Developer
Common Functions

- CONCAT or || = concatenation
- INITCAP = capitalizes first character of each word in string
- LENGTH = number of characters in string
- LOWER or UPPER = changes case of word
- TRIM = remove spaces or specific characters from string
  - LTRIM and RTRIM are versions of TRIM
- LPAD or RPAD = pad to the left or right with spaces or other characters
- REPLACE = replace character in a string with a different character or nothing at all
- SUBSTR = pulls out a specific string of characters
Common Functions

- **COUNT** = total # of records for a given condition
- **ROWNUM** = adds a row number to records in your result set and can be sorted
- **ABS** = returns absolute value
- **AVG, MAX, MIN, and SUM** = aggregate functions that require a grouping
- **ROUND** = rounds a number to specified decimal place or whole number
- **TRUNC** = truncate numbers and dates
- **TO_CHAR** = converts a number or date to a string
- **MONTHS_BETWEEN** = # of months between two dates
Common Functions

- **TO_DATE** = makes a string into a date
- **TO_NUMBER** = converts a string into a number
- **CAST** = converts to a different datatype
- **RANK** = ranks results within a group
- **LISTAGG** = adds values of a specific column within a group
- **CASE** = if-then-else statement (when <exp> then <value>)
- **DECODE** = another if-then-else statement
- **NVL & NVL2** = if value null then replace with
- **NULLIF** = compares two expressions, returns null if matched
- **COALESCE** = returns first non-null value of list
- **USER** = returns the user_id of the session
One more...
Future Topics

- GROUP BY, HAVING clauses
- Subqueries
- WITH (for temporary, session tables)
- PIVOT
- UPDATE, INSERT, and DELETE statements
- Metadata tables
- Advanced Oracle functions
- Custom Banner functions
- Indexes
- Declaring variables in a script
- CREATE statements
Next Steps...

- Find out what kind of access you have
- **NEVER, NEVER, NEVER** write new code from scratch in **PROD** … develop in a non-prod environment first
- Find and attend online training
- Just start writing code in any kind of SQL environment *(except in production)*
- Ask questions
- Take advantage of every opportunity
Resources

- LinkedIn Learning
  - SQL Essential Training
  - Learning SQL Programming
- Coursera.org
- Learning SQL on Amazon
- SQL Cookbook on Amazon
- Tech on the Net Tutorial
- Oracle functions w/examples
- Lehigh’s Data Cookbook
Thank you!

If you have any questions, feel free to reach out to us at ludba@lehigh.edu