ORACLE SQL & BRIDGING FROM POSTGRESQL
Key Differences & Core Syntax, Data Types - DUAL Table - ROWNUM Pseudo-column *(Oracle Specific)*, NULL Handling - Conditional Expressions *(Practice in Oracle)*, Comments: *Exercises*

Transitional SQL

May 27, 2025

Contents

1 Dataset

The following dataset will be used for the exercises. Ensure these tables are created and populated in your Oracle SQL environment.

```sql
-- Ensure clean environment (optional, run if tables might exist)
BEGIN
   EXECUTE IMMEDIATE 'DROP TABLE ProductSales';
EXCEPTION WHEN OTHERS THEN IF SQLCODE != -942 THEN RAISE; END IF;
END;
/
BEGIN
   EXECUTE IMMEDIATE 'DROP TABLE ProductCatalog';
EXCEPTION WHEN OTHERS THEN IF SQLCODE != -942 THEN RAISE; END IF;
END;
/
BEGIN
   EXECUTE IMMEDIATE 'DROP TABLE EmployeeRoster';
EXCEPTION WHEN OTHERS THEN IF SQLCODE != -942 THEN RAISE; END IF;
END;
/

-- Table for demonstrating various data types, NULLs, and for ROWNUM
CREATE TABLE EmployeeRoster (
    employeeId NUMBER(6) PRIMARY KEY,
    firstName VARCHAR2(50),
    lastName VARCHAR2(50),
    email VARCHAR2(100) UNIQUE,
    phoneNumber VARCHAR2(20),
    hireDate DATE, -- In Oracle, DATE stores date and time
    jobTitle VARCHAR2(50),
    salary NUMBER(10, 2),
    commissionRate NUMBER(4, 2), -- Can be NULL
    managerId NUMBER(6),
    departmentName VARCHAR2(50),
    bio NVARCHAR2(100) -- For NVARCHAR2 example
);

-- Table for DUAL, DECODE/CASE, and different timestamp types
CREATE TABLE ProductCatalog (
    productId NUMBER(5) PRIMARY KEY,
    productName VARCHAR2(100),
    productCategory VARCHAR2(50),
    unitPrice NUMBER(8, 2),
    supplierInfo NVARCHAR2(100), -- For international supplier names
    lastStockCheck TIMESTAMP(3), -- Timestamp with 3 fractional seconds
    nextShipmentDue TIMESTAMP(0) WITH TIME ZONE, -- Timestamp with 0 fractional seconds, with
     time zone
    localEntryTime TIMESTAMP WITH LOCAL TIME ZONE, -- Stores in DB timezone, displays in
     session timezone
    notes VARCHAR2(100) -- Can be NULL, to show NVL/COALESCE
);

-- A smaller table to illustrate ROWNUM behavior more clearly with ORDER BY
CREATE TABLE ProductSales (
    saleId NUMBER PRIMARY KEY,
    productSold VARCHAR2(50),
    saleAmount NUMBER(10,2),
    saleDate DATE
);

-- Comments: Example of single-line and multi-line comments
/*
This is a multi-line comment.
These tables are designed for the Oracle SQL transitional course.
The EmployeeRoster table includes an NVARCHAR2 column 'bio' to store
employee biographies, potentially in multiple languages.
The ProductCatalog table uses various TIMESTAMP types to track product-related timings.
*/

-- Populate EmployeeRoster
-- Note: TO_DATE without time component defaults to 00:00:00 for the time part
```

```sql
INSERT INTO EmployeeRoster (employeeId, firstName, lastName, email, phoneNumber, hireDate, jobTitle, salary, commissionRate, managerId, departmentName, bio)
VALUES (100, 'Steven', 'King', 'SKING', '515.123.4567', TO_DATE('2003-06-17', 'YYYY-MM-DD'), 'President', 24000, NULL, NULL, 'Executive', N'Oversees all operations. スティーブン');
INSERT INTO EmployeeRoster (employeeId, firstName, lastName, email, phoneNumber, hireDate, jobTitle, salary, commissionRate, managerId, departmentName, bio)
VALUES (101, 'Neena', 'Kochhar', 'NKOCHHAR', '515.123.4568', TO_DATE('2005-09-21', 'YYYY-MM-DD'), 'Administration VP', 17000, NULL, 100, 'Administration', N'Manages admin staff. □□□□');
INSERT INTO EmployeeRoster (employeeId, firstName, lastName, email, phoneNumber, hireDate, jobTitle, salary, commissionRate, managerId, departmentName, bio)
VALUES (102, 'Lex', 'De Haan', 'LDEHAAN', '515.123.4569', TO_DATE('2001-01-13', 'YYYY-MM-DD'), 'Administration VP', 17000, 0.15, 100, 'Administration', N'Also an Admin VP. レックス');
INSERT INTO EmployeeRoster (employeeId, firstName, lastName, email, phoneNumber, hireDate, jobTitle, salary, commissionRate, managerId, departmentName, bio)
VALUES (103, 'Alexander', 'Hunold', 'AHUNOLD', '590.423.4567', TO_DATE('2006-01-03', 'YYYY-MM-DD'), 'Programmer', 9000, 0.10, 102, 'IT', N'Develops software. αλέξανδρος');
INSERT INTO EmployeeRoster (employeeId, firstName, lastName, email, phoneNumber, hireDate, jobTitle, salary, commissionRate, managerId, departmentName, bio)
VALUES (104, 'Bruce', 'Ernst', 'BERNST', '590.423.4568', TO_DATE('2007-05-21', 'YYYY-MM-DD'), 'Programmer', 6000, 0.10, 103, 'IT', NULL); -- Null bio
INSERT INTO EmployeeRoster (employeeId, firstName, lastName, email, phoneNumber, hireDate, jobTitle, salary, commissionRate, managerId, departmentName, bio)
VALUES (107, 'Diana', 'Lorentz', 'DLORENTZ', '590.423.5567', TO_DATE('2007-02-07', 'YYYY-MM-DD'), 'Finance Manager', 12000, 0.20, 101, 'Finance', N'Financial planning.');
INSERT INTO EmployeeRoster (employeeId, firstName, lastName, email, phoneNumber, hireDate, jobTitle, salary, commissionRate, managerId, departmentName, bio)
VALUES (114, 'Den', 'Raphaely', 'DRAPHEALY', '515.127.4561', TO_DATE('2002-12-07', 'YYYY-MM-DD'), 'Purchasing Manager', 11000, NULL, 100, 'Purchasing', N'Procurement. '□□);

-- Populate ProductCatalog
-- Setting session time zone for predictable TIMESTAMP WITH LOCAL TIME ZONE insertion for demo
ALTER SESSION SET TIME_ZONE = 'America/New_York'; -- Example: EST/EDT

INSERT INTO ProductCatalog (productId, productName, productCategory, unitPrice, supplierInfo, lastStockCheck, nextShipmentDue, localEntryTime, notes)
VALUES (1, 'Oracle Database 19c', 'Software', 5000, N'Oracle Corp. USA', TIMESTAMP '2023-10-01 10:00:00.123', TIMESTAMP '2023-11-15 09:00:00 -05:00', SYSTIMESTAMP, 'Enterprise Edition');
INSERT INTO ProductCatalog (productId, productName, productCategory, unitPrice, supplierInfo, lastStockCheck, nextShipmentDue, localEntryTime, notes)
VALUES (2, 'PostgreSQL 15', 'Software', 0, N'PG Global Dev Group', TIMESTAMP '2023-10-05 11:30:00.456', TIMESTAMP '2023-10-20 14:00:00 +02:00', SYSTIMESTAMP - INTERVAL '1' DAY, NULL);
INSERT INTO ProductCatalog (productId, productName, productCategory, unitPrice, supplierInfo, lastStockCheck, nextShipmentDue, localEntryTime, notes)
VALUES (3, 'SQL Developer Tool', 'Utility', 100, N'DevTools Inc. (Canada)', TIMESTAMP '2023-09-20 15:00:00.789', NULL, SYSTIMESTAMP - INTERVAL '2' DAY, 'Cross-RDBMS');
INSERT INTO ProductCatalog (productId, productName, productCategory, unitPrice, supplierInfo, lastStockCheck, nextShipmentDue, localEntryTime, notes)
VALUES (4, 'Advanced Java Book', 'Book', 75, N'Tech Books GmbH (Germany) - Bücher', TIMESTAMP '2023-10-10 08:00:00.000', FROM_TZ(TIMESTAMP '2023-11-01 10:00:00.000', 'UTC'), SYSTIMESTAMP - INTERVAL '3' DAY, 'Includes Oracle examples');
INSERT INTO ProductCatalog (productId, productName, productCategory, unitPrice, supplierInfo, lastStockCheck, nextShipmentDue, localEntryTime, notes)
VALUES (5, 'Unicode Keyboard', 'Hardware', 50, N' 全球配件 (Global Accessories)', TIMESTAMP '2023-10-12 16:45:00.999', NULL, SYSTIMESTAMP - INTERVAL '4' DAY, 'Supports various languages');

-- Populate ProductSales
INSERT INTO ProductSales (saleId, productSold, saleAmount, saleDate) VALUES (1, 'Product A', 100.50, TO_DATE('2023-01-15', 'YYYY-MM-DD'));
INSERT INTO ProductSales (saleId, productSold, saleAmount, saleDate) VALUES (2, 'Product B', 250.00, TO_DATE('2023-01-10', 'YYYY-MM-DD'));
INSERT INTO ProductSales (saleId, productSold, saleAmount, saleDate) VALUES (3, 'Product C', 75.25, TO_DATE('2023-02-01', 'YYYY-MM-DD'));
INSERT INTO ProductSales (saleId, productSold, saleAmount, saleDate) VALUES (4, 'Product A', 100.50, TO_DATE('2023-02-05', 'YYYY-MM-DD'));
INSERT INTO ProductSales (saleId, productSold, saleAmount, saleDate) VALUES (5, 'Product D',
    500.00, TO_DATE('2023-02-10', 'YYYY-MM-DD'));
COMMIT;
```

Listing 1: Oracle SQL Dataset for Core Syntax, Data Types, and Control Flow Exercises

2 Meanings, Values, Relations, and Advantages

2.1 Exercise 1.1: Understanding Oracle Data Types & Bridging from PostgreSQL

1. **VARCHAR2 vs. NVARCHAR2:**
   - Explain the core difference between `VARCHAR2` and `NVARCHAR2` in Oracle.
   - The `EmployeeRoster` table has `firstName` (`VARCHAR2`) and `bio` (`NVARCHAR2`). Why might `bio` be `NVARCHAR2` while `firstName` (in many Western contexts) might be `VARCHAR2`?
   - In PostgreSQL, you commonly use `VARCHAR` or `TEXT`. How does `VARCHAR2` relate, and what Oracle-specific considerations are there for character data?
2. **NUMBER Type:**
   - Oracle's `NUMBER` type is used for `employeeId` and `salary` in `EmployeeRoster`. Illustrate how `NUMBER` definition (`NUMBER(6)` vs `NUMBER(10,2)`) achieves this.
   - What PostgreSQL types (e.g., `INTEGER`, `NUMERIC`) would correspond to these uses? What is an advantage of Oracle's unified `NUMBER` type?
3. **DATE Type:**
   - Retrieve `employeeId` and `hireDate` for 'Steven King'. Note the format.
   - PostgreSQL's `DATE` type stores only date. Oracle's `DATE` stores date and time. What PostgreSQL type is Oracle's `DATE` most analogous to? How could this difference impact data migration or queries if not handled carefully?
4. **TIMESTAMP Variations:**
   - From `ProductCatalog`, select `productName`, `lastStockCheck` (`TIMESTAMP`), `nextShipmentDue` (`TIMESTAMP WITH TIME ZONE`), and `localEntryTime` (`TIMESTAMP WITH LOCAL TIME ZONE`) for 'Oracle Database 19c'.
   - Briefly explain the advantage of each `TIMESTAMP` variant chosen for these columns. (You may want to run `ALTER SESSION SET NLS_TIMESTAMP_FORMAT = 'YYYY-MM-DD HH24:MI:SS.FF';` and `ALTER SESSION SET NLS_TIMESTAMP_T` `= 'YYYY-MM-DD HH24:MI:SS.FF TZR';` for clarity).

2.2 Exercise 1.2: DUAL Table and NULL Handling (NVL, NVL2, COALESCE)

1. **DUAL Table:**
   - What is the `DUAL` table in Oracle? Give two common use cases.
   - In PostgreSQL, `SELECT 1+1;` works. How do you achieve this in Oracle and why is `DUAL` needed?

2. **NVL Function:**
   - Display `employeeId`, `firstName`, `salary`, `commissionRate`, and a "Guaranteed Pay" which is `salary + (salary * commissionRate)`. If `commissionRate` is NULL, it should be treated as 0. Use `NVL`.
   - How does `NVL(expr1, expr2)` compare to PostgreSQL's `COALESCE(expr1, expr2)`?
3. **NVL2 Function:**
   - Display `employeeId`, `firstName`, and a `commissionStatus`. If `commissionRate` is NOT NULL, `commissionStatus` should be 'Eligible for Commission Bonus'. If `commissionRate` IS NULL, it should be 'Salary Only'. Use `NVL2`.
   - How would you achieve the `NVL2` logic using standard SQL constructs known from PostgreSQL (like `CASE`)?
4. **COALESCE Function:**
   - From `ProductCatalog`, display `productId`, `productName`, and the `notes`. If `notes` is NULL, show 'No additional notes'. If `notes` is NULL and `supplierInfo` also happens to be NULL (not in current data, but imagine), show 'Critical info missing'. Use `COALESCE`.

2.3 Exercise 1.3: Conditional Logic (DECODE, CASE) & Comments

1. **DECODE vs. CASE:**
   - What is a key syntactical difference between Oracle's `DECODE` function and the standard `CASE` expression when performing multiple comparisons?
   - Which is generally more readable and flexible for complex conditions?
2. **DECODE Function:**
   - Using `DECODE` on `EmployeeRoster`, display `firstName`, `jobTitle`. Add a new column `jobLevel`. If `jobTitle` is 'President', `jobLevel` is 'Top Tier'. If 'Administration VP' or 'Finance Manager', it's 'Mid Tier'. If 'Programmer', it's 'Staff'. Otherwise, 'Other'.
3. **CASE Expression:**
   - Rewrite the query from (1.3.2) using a `CASE` expression (use a searched `CASE` for clarity).
   - From `ProductCatalog`, display `productName`, `unitPrice`, and a `priceTag`. If `unitPrice` = 0, `priceTag` is 'Free'. If `unitPrice` > 0 AND `unitPrice` <= 100, `priceTag` is 'Affordable'. If `unitPrice` > 100, `priceTag` is 'Premium'. Use a `CASE` expression.
4. **Comments:**

- Add a single-line comment above your `CASE` expression query explaining its purpose.
- Add a multi-line comment at the beginning of your SQL script file for this exercise set, stating the Oracle concepts being practiced.

2.4 Exercise 1.4: ROWNUM Pseudo-column

1. **ROWNUM Basics:**
   - What is `ROWNUM` in Oracle? When is its value assigned to a row in a query's execution?
   - How does `ROWNUM` fundamentally differ from PostgreSQL's `LIMIT` clause in behavior, especially concerning `ORDER BY`?
2. **Top-N Query:**
   - Select the `firstName`, `lastName`, and `salary` of the 3 employees with the highest salaries from `EmployeeRoster`. Ensure `ROWNUM` is used correctly for this.
3. **Pagination Emulation (Conceptual):**
   - Explain how you would select the employees who are, say, the 4th and 5th highest paid (i.e., rows 4-5 in a list sorted by salary descending). You must use `ROWNUM`.

3 Disadvantages and Pitfalls

3.1 Exercise 2.1: Data Type Pitfalls and Misunderstandings

1. **VARCHAR2 Size & Semantics:** An `EmployeeRoster` `firstName` column is `VARCHAR2(10 BYTE)`. What happens if you try to insert 'Christophe' (10 chars, 10 bytes in ASCII)? What if you try to insert 'René' (4 chars, but 'é' can be 2 bytes in UTF8)? What is the pitfall if `NLS_LENGTH_SEMANTICS` is `BYTE` when dealing with multi-byte characters?

2. **NUMBER Precision/Scale:**

   - If `salary` in `EmployeeRoster` was defined only as `NUMBER` (no precision/scale) and you inserted `12345.678912345`, what would be stored? What's a potential pitfall of omitting precision/scale for financial data?
   - If `commissionRate` is `NUMBER(4,2)` and you attempt to insert `0.125` or `10.50`. What happens in each case? What if you try to insert `123.45`?

3. **Oracle DATE Time Component:** A PostgreSQL user accustomed to `DATE` being date-only inserts `TO_DATE('2023-11-10', 'YYYY-MM-DD')` into `hireDate` (Oracle `DATE`). They later run `SELECT * FROM EmployeeRoster WHERE hireDate = TO_DATE('2023-11-10 10:00:00', 'YYYY-MM-DD HH24:MI:SS');`. Will they find the record? Why or why not? What's the pitfall?

4. **TIMESTAMP WITH LOCAL TIME ZONE (TSLTZ):** `localEntryTime` in `ProductCatalo` is `TIMESTAMP WITH LOCAL TIME ZONE`.

   - Session A (Time Zone 'America/New_York') inserts `TIMESTAMP '2023-11-10 10:00:00 America/New_York'`.
   - Session B (Time Zone 'Europe/London') queries this exact row. What time will Session B see (conceptually, considering typical UTC offsets)?
   - What is a potential pitfall if the database's `DBTIMEZONE` is different from the application server's OS time zone, and TSLTZ data is inserted using `SYSTIMESTAMP` without explicit time zone specification?

3.2 Exercise 2.2: NULL Handling Function Caveats

1. **NVL Type Conversion:** What happens if you use `NVL(salary, 'Not Available')` where `salary` is `NUMBER(10,2)`? Why is this a pitfall? How should it be corrected if the goal is a string output?

2. **NVL2 Type Mismatch:** Consider `NVL2(hireDate, SYSDATE + 7, 'Not Hired Yet')`. `hireDate` is `DATE`. What is the likely data type of the result if `hireDate` is NOT NULL? What if it IS NULL? What's the potential issue and how can Oracle try to resolve it (possibly leading to errors)?

3. **COALESCE Argument Evaluation:** While `COALESCE` returns the first non-NULL expression, all expressions provided to it must be of data types that are implicitly convertible to a common data type, determined by the first non-NULL expression. What error might occur with `COALESCE(numericColumn, dateColumn, 'textFallback')` if `numericColumn` is NULL but `dateColumn` is not?

3.3 Exercise 2.3: DECODE and ROWNUM Logic Traps

1. **DECODE's NULL Handling:** `DECODE(colA, colB, 'Match', 'No Match')`. If both `colA` and `colB` are `NULL`, what does this return? How does this differ from `CASE WHEN colA = colB THEN 'Match' ELSE 'No Match' END`? When could `DECODE`'s behavior be a pitfall?

2. **ROWNUM for Pagination - Incorrect Attempt:** A developer wants to display the 3rd and 4th products from `ProductCatalog` (in order of `productId`). They write:

   ```sql
   SELECT productName FROM ProductCatalog WHERE ROWNUM BETWEEN 3 AND 4
      ORDER BY productId;
   ```

   Why will this query return no rows?

3. **ROWNUM with ORDER BY - Misconception:** What is the output of the following query? Is it guaranteed to be the two products whose names are last alphabetically? Explain.

   ```sql
   SELECT productName, ROWNUM FROM ProductCatalog WHERE ROWNUM <= 2 ORDER
       BY productName DESC;
   ```

4 Contrasting with Inefficient Common Solutions

4.1 Exercise 3.1: Suboptimal Logic vs. Oracle SQL Efficiency

1. **Client-Side NULL Handling:** A developer fetches `firstName` and `commissionRate` from `EmployeeRoster`. In their application code (e.g., Java/Python), they loop through results: if `commissionRate` is null, they display "$0.00", otherwise they display the actual rate.

   - Show the efficient Oracle SQL way to produce a `commissionDisplay` column directly using an Oracle NULL handling function.
   - What is the loss of advantage (e.g., performance, network traffic) with the client-side approach?

2. **Client-Side Conditional Logic:** For each product in `ProductCatalog`, if `productCategory` is 'Software', display 'Digital Good'. If 'Hardware', display 'Physical Good'. Otherwise, 'Misc Good'. This logic is currently in client code.

   - Demonstrate the efficient Oracle SQL way using a `CASE` expression.
   - Why is performing this categorization in SQL generally better than in client code for reporting?

4.2 Exercise 3.2: Inefficient ROWNUM Usage and DUAL Misconceptions

1. **Inefficient DUAL Usage:** A process needs to log the current timestamp and the current user performing an action. The developer writes:

```sql
-- Get timestamp
SELECT SYSTIMESTAMP FROM DUAL; -- Result captured by app
-- Get user
SELECT USER FROM DUAL;       -- Result captured by app
```

   Show the efficient way. What Oracle value is lost by the inefficient approach?

2. **Incorrect Top-N with ROWNUM:** To find the 3 cheapest products *that are not free* from `ProductCatalog`, a developer writes:

```sql
SELECT productName, unitPrice
FROM ProductCatalog
WHERE unitPrice > 0 AND ROWNUM <= 3 -- Attempt to filter non-free
    first, then take top 3
ORDER BY unitPrice ASC;
```

   Explain why this is not guaranteed to give the 3 overall cheapest non-free products. Present the efficient, correct Oracle-idiomatic way.

5 Hardcore Combined Problem

5.1 Exercise 4.1: Multi-Concept Oracle Challenge for "Employee Performance Review Prep"

Scenario: Management needs a preliminary report for performance reviews. The report should identify the top 2 longest-serving 'Programmer' employees from the 'IT' department. For these employees, provide a "Review Focus" and details about their bio and tenure.

Requirements:

1. **Selection:** Target 'Programmer' employees in the 'IT' department only.

2. **Output Columns:**

   - `employeeId` (NUMBER)
   - `employeeName` (VARCHAR2, format: 'LastName, FirstName')
   - `jobTitle` (VARCHAR2)
   - `department` (VARCHAR2)
   - `hireDateDisplay` (VARCHAR2, formatted as 'Month DD, YYYY', e.g., 'January 03, 2006')
   - `yearsOfService` (NUMBER, calculated to one decimal place from `hireDate` to `SYSDATE`. Use `MONTHS_BETWEEN` and `DUAL` for `SYSDATE` if needed in calculation context, though `SYSDATE` can be used directly).
   - `bioExtract` (NVARCHAR2: If `bio` is not NULL, show the first 30 characters of `bio` followed by '...'. If `bio` is NULL, display 'No Bio on File'. Use `NVL` or `COALESCE` and string functions).
   - `reviewFocus` (VARCHAR2):
     - Use a `CASE` expression.
     - If `commissionRate` IS NOT NULL, focus is 'Sales & Technical Skills Review'.
     - Else (if `commissionRate` IS NULL):
       - Use `DECODE` on `managerId`. If `managerId` is 102, focus is 'Project Leadership Potential'.
       - Otherwise (for other managers or NULL managerId for programmers), focus is 'Core Technical Deep Dive'.

3. **Top-N Logic:** The final output must be strictly limited to the top 2 longest-serving employees (earliest `hireDate`) based on the above criteria. Use `ROWNUM` correctly for this.

4. **Comments:** Include a brief multi-line comment explaining the report's purpose and a single-line comment for the `ROWNUM` filtering logic.

5. **DUAL Table (Implicit/Explicit):** Use of `SYSDATE` implicitly involves concepts related to `DUAL`'s role in providing such values.

Bridging from PostgreSQL: This problem involves concepts like string manipulation (`SUBSTR`, concatenation), date calculations (`MONTHS_BETWEEN` vs. PostgreSQL age/interval functions), conditional logic (`CASE` is similar, `DECODE` is new), NULL handling (`NVL`/`COALESCE` vs. PG `COALESCE`), and Top-N queries (`ROWNUM` vs. PG `LIMIT`).