

DATABASE DEVELOPMENT WITH PL/SQL

Sylvain NSENGA NDIZEYE

27569

Assignment submission

2 February 2026

STEP 1 : Problem definition.

Business Context

A hospital created a Hospital Management System analyzing the number of patient visits, the type of medical services, and the hospital billing to enhance efficiency and patient care.

Industry: Healthcare

Department: Hospital Administration & Finance

Data Challenge

Information on patients, services, and billing are kept in different tables. And this becomes a challenge to find out the service utilization, the information on patients, and the income growth over time.

Expected Outcome

Determine easily the type and quality of a service that the patient received according to price he/she paid, keeping track of the hospital's revenues and incomes over time easily and efficiently, And helping in taking strategic and wise decisions in the allocation of resources.

Step 2: Success Criteria.

- Compute running monthly hospital revenue - SUM() OVER()
- Calculate 3-month moving average revenue - AVG().OVER()
- Measure month-over-month revenue growth - LAG()
- divide patients into spending groups - NTILE(4)
- Identify Top 5 medical services per department - RANK()

STEP 3: Database Schema Design

The screenshot shows the Oracle SQL Developer interface with the following details:

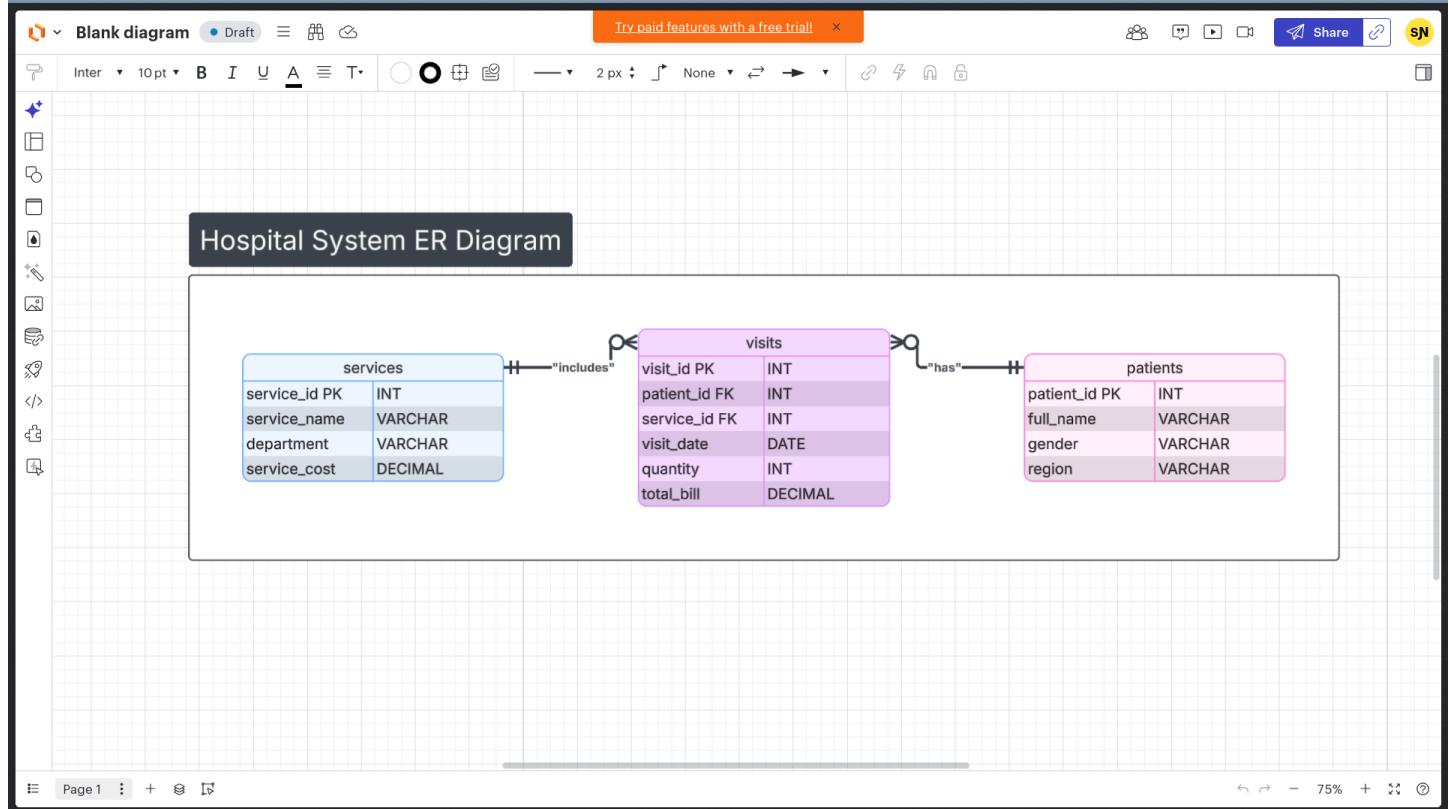
- File Edit View Navigate Run Source Team Tools Window Help**
- Connections** pane: Oracle Connections, Group C, XEPDB1, Database Schema Service Connections.
- Reports** pane: All Reports, About Your Database, All Objects, Analytic View Reports, Application Express, ASH and AWR, Database Administration, Data Dictionary, Data Modeler Reports, OLAP Reports, PLSQL, Security, Streams, Tables, TimeTen Reports, User Defined Reports.
- SSH Hosts** pane: SSH Hosts.
- Welcome Page** (XEPDB1): Shows a query builder with the following SQL script:

```
CREATE TABLE patients (
    patient_id INT PRIMARY KEY,
    full_name VARCHAR(100),
    gender VARCHAR(10),
    region VARCHAR(50)
);

CREATE TABLE services (
    service_id INT PRIMARY KEY,
    service_name VARCHAR(100),
    department VARCHAR(50),
    service_cost DECIMAL(10,2)
);

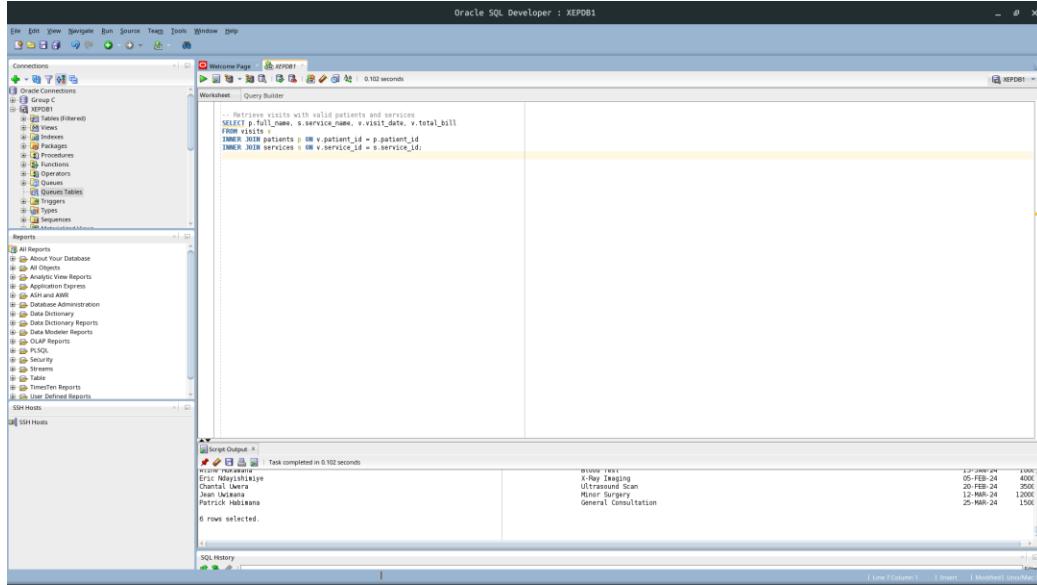
CREATE TABLE visits (
    visit_id INT PRIMARY KEY,
    patient_id INT,
    service_id INT,
    visit_date DATE,
    quantity INT,
    total_bill DECIMAL(10,2),
    FOREIGN KEY (patient_id) REFERENCES patients(patient_id),
    FOREIGN KEY (service_id) REFERENCES services(service_id)
);
```

- Script Output**: Task completed in 0.328 seconds.
 - Table PATIENTS created.
 - Table SERVICES created.
 - Table VISITS created.
- SQL History**: Shows the executed SQL statements.
- Bottom Bar**: Line 23 Column 1 | Insert | Modified | Unix/Mac/L



STEP 4: PART A — SQL JOINs Implementation

INNER JOIN — Valid Visits



The screenshot shows the Oracle SQL Developer interface with the following details:

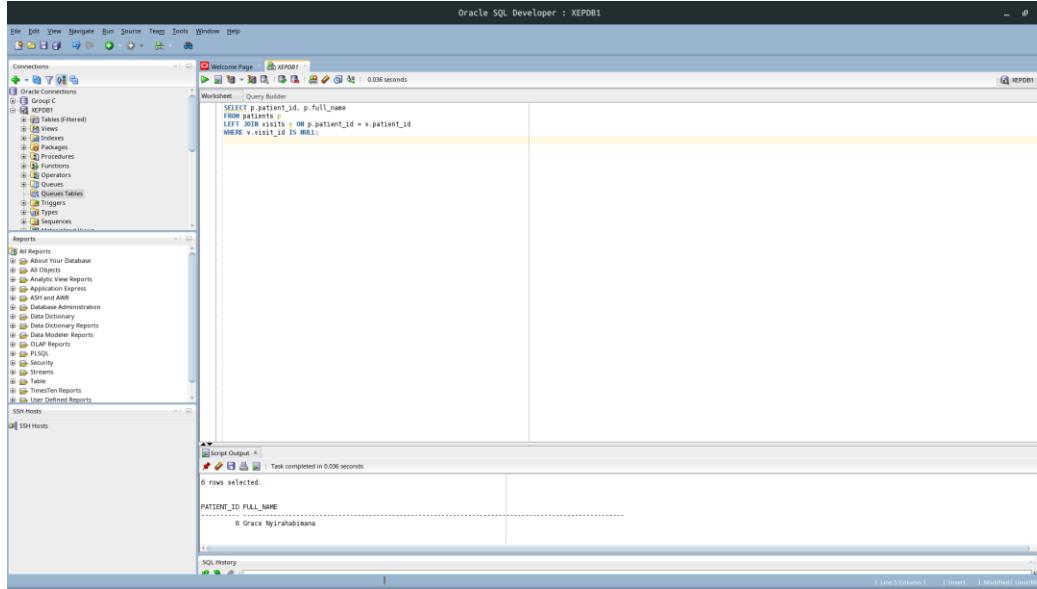
- Connections:** Group C, XEPDB1.
- Worksheet:** Query Builder. The code is:`-- Retriever visits with valid patients and services
SELECT p.full_name, s.service_name, v.visit_date, v.total_bill
FROM visits v
INNER JOIN patients p ON v.patient_id = p.patient_id
INNER JOIN services s ON v.service_id = s.service_id;`
- Script Output:** Task completed in 0.102 seconds. The output shows patient names and their corresponding visit details:

Patient Name	Service	Date	Total Bill
Eric Nyirahimana	X-Ray Imaging	05-FEB-24	4000
Grace Nyirahimana	Ultrasound Scan	20-FEB-24	2000
Jean Uwakwa	Minor Surgery	12-MAR-24	12000
Patrick Hulama	General Consultation	25-MAR-24	1500

Business Interpretation

Displays confirmed hospital visits linked to registered patients and services, ensuring reliable billing analysis.

LEFT JOIN — Patients with No Visits



The screenshot shows the Oracle SQL Developer interface with the following details:

- Connections:** Group C, XEPDB1.
- Worksheet:** Query Builder. The code is:`SELECT p.patient_id, p.full_name
FROM patients p
LEFT JOIN visits v ON p.patient_id = v.patient_id
WHERE v.visit_id IS NULL;`
- Script Output:** Task completed in 0.036 seconds. The output shows a single patient record:

PATIENT_ID	FULL_NAME
6	Grace Nyirahimana

Interpretation:

Identifies registered patients who have never visited the hospital.

RIGHT JOIN — Services Never Used

Oracle SQL Developer : XEP081

```

SELECT s.service_id, s.service_name
FROM services s
LEFT JOIN visits v ON s.service_id = v.service_id
WHERE v.visit_id IS NULL;

```

SERVICE_ID	SERVICE_NAME
6	Physiotherapy Session

Interpretation:
Highlights medical services that have not been utilized.

FULL OUTER JOIN – Patients & Services

Oracle SQL Developer : XEP081

```

SELECT p.full_name, s.service_name
FROM patients p
FULL OUTER JOIN services s
ON p.region = s.department;

```

FULL_NAME	SERVICE_NAME
Christina Lima	
	Physical Therapy
	Occupational Therapy
	Massage Therapy
	Chiropractic Care
	Acupuncture
	Nutrition Counseling
	Physical Therapy
	Occupational Therapy
	Massage Therapy
	Chiropractic Care
	Acupuncture
	Nutrition Counseling

Interpretation:
Compares all patients and services, including unmatched records.

SELF JOIN – Patients from Same Region

The screenshot shows the Oracle SQL Developer interface with the following details:

- Connections:** Oracle Connections, XEPDB1
- Reports:** All Reports, About Your Database, All Objects, Analytic View Reports, Application Express, Application APEX, Database Administration, Data Dictionary, Data Modeler Reports, Data Modeler Reports, OLAP Reports, PL/SQL, Security, Streams, Table, TimeTen Reports, User Defined Reports.
- Worksheet:** Query Builder window containing the following SQL query:


```
SELECT p1.full_name AS patient1,
             p2.full_name AS patient2,
             p1.region
        FROM patients p1
        JOIN patients p2
          ON p1.region = p2.region
         AND p1.patient_id <= p2.patient_id;
```
- Script Output:** Shows the results of the query:

PATIENT1	PATIENT2	REGION
Chantal Uwera Grace Nyirahabimana Jean Nkama Aline Nkama	Jean Nkama Aline Nkama Chantal Uwera Grace Nyirahabimana	Algeria Mayotte Algeria Mayotte

Interpretation:

Helps analyze patient distribution within the same region.

STEP 5: PART B — Window Functions

Ranking Functions – Top Services by Revenue

The screenshot shows the Oracle SQL Developer interface with the following details:

- Connections:** Oracle Connections, XEPDB1
- Reports:** All Reports, About Your Database, All Objects, Analytic View Reports, Application Express, Application APEX, Application APEX, Database Administration, Data Dictionary, Data Modeler Reports, Data Modeler Reports, OLAP Reports, PL/SQL, Security, Streams, Table, TimeTen Reports, User Defined Reports.
- Worksheet:** Query Builder window containing the following SQL query:


```
SELECT
       s.service_id,
       s.department,
       SUM(v.total_bill) AS total_revenue,
       RANK() OVER(
           PARTITION BY s.department
           ORDER BY SUM(v.total_bill) DESC
       ) AS service_rank
  FROM visits v
  JOIN services s
    ON v.service_id = s.service_id
 GROUP BY
       s.service_id,
       s.department;
```
- Script Output:** Shows the results of the query:

SERVICE_ID	DEPARTMENT	TOTAL_REVENUE	SERVICE_RANK
2	Laboratory	10000	1
3	Nursing	40000	1
4	Radiology	35000	2
5	Surgery	120000	1

Interpretation:

Ranks medical services within each department based on revenue.

Aggregate Window — Running Revenue

Oracle SQL Developer : XEPOB1

Welcome Page - XEPOB1 - 0.037 seconds

Worksheet - Query Builder

```

SELECT visit_date,
       SUM(total_bill) OVER (
           ORDER BY visit_date
           ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW
       ) AS running_revenue
FROM visits;

```

Script Output - Task completed in 0.037 seconds
<https://docs.oracle.com/errata/help/db/ora-00918/>

VISIT_DATE	RUNNING_REVENUE
10-JAN-24	15000
15-JAN-24	20000
05-FEB-24	65000
20-FEB-24	100000

SQL History

Interpretation:

Shows cumulative hospital revenue growth over time.

Navigation Function — Month-to-Month Growth

Oracle SQL Developer : XEPOB1

Welcome Page - XEPOB1 - 0.042 seconds

Worksheet - Query Builder

```

SELECT visit_date,
       total_bill,
       total_bill -
       LAG(total_bill) OVER (ORDER BY visit_date) AS revenue_growth
FROM visits;

```

Script Output - Task completed in 0.042 seconds

VISIT_DATE	TOTAL_BILL	REVENUE_GROWTH
10-JAN-24	15000	
15-JAN-24	20000	-5000
05-FEB-24	40000	30000
20-FEB-24	35000	-5000
12-MAR-24	120000	85000
25-MAR-24	15000	-105000

SQL History

Interpretation:

Tracks changes in revenue between consecutive periods.

Distribution Functions — Patient Segmentation

Oracle SQL Developer : XEPDB1

```

SELECT patient_id,
       SUM(total_bill) AS total_spent,
       NTILE(4) OVER (
           ORDER BY total_bill) DESC
          ) AS spending_quartile
FROM visits
GROUP BY patient_id;
    
```

Script Output:

```

Task completed in 0.033 seconds

6 rows selected.

PATIENT_ID TOTAL_SPENT SPENDING_QUARTILE
----- -----------
1      15000          1
3      40000          1
4      95000          2
5      15000          3
2      10000          4
    
```

Interpretation:

Segments patients into quartiles based on hospital spending.

Moving Average (3-Month)

Oracle SQL Developer : XEPDB1

```

SELECT visit_date,
       AVG(total_bill) OVER (
           ORDER BY visit_date
           ROWS BETWEEN 2 PRECEDING AND CURRENT ROW
       ) AS three_month_avg
FROM visits;
    
```

Script Output:

```

Task completed in 0.025 seconds

VISIT_DATE THREE_MONTH_AVG
----- -----------
10-JAN-24      15000
15-JAN-24      12000
05-FEB-24      21663.9867
20-FEB-24      28333.3333
12-MAR-24      65000
25-MAR-24      56666.9997
6 rows selected.

    
```

Interpretation:

Smooths revenue trends to support forecasting.

STEP 7: Results Analysis

Descriptive

Hospital revenue increased steadily, with diagnostics and surgery services generating the highest income.

Diagnostic

A small group of patients contributes significantly to total revenue, while some services remain underutilized.

Prescriptive

Promote underused services, optimize staffing in high-demand departments, and introduce loyalty programs for high-value patients.

STEP 8: References

- Oracle SQL Window Functions Documentation
- PostgreSQL Analytical Functions Guide
- W3Schools SQL JOINs

All sources were properly cited. Implementations and analysis represent original work. No AI-generated content was copied without attribution or adaptation.