

# **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

Oracle SQL Developer : XEPDB1

File Edit View Navigate Run Source Team Tools Window Help

Connections

Oracle Connections

Group C

XEPDB1

Database Schema Service Connections

Reports

All Reports

About Your Database

All Objects

Analytic View Reports

Application Express

ASH and AWR

Database Administration

Data Dictionary

Data Dictionary Reports

Data Modeler Reports

OLAP Reports

PLSQL

Security

Streams

Table

TimesTen Reports

User Defined Reports

SSH Hosts

SSH Hosts

Worksheet

Query Builder

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

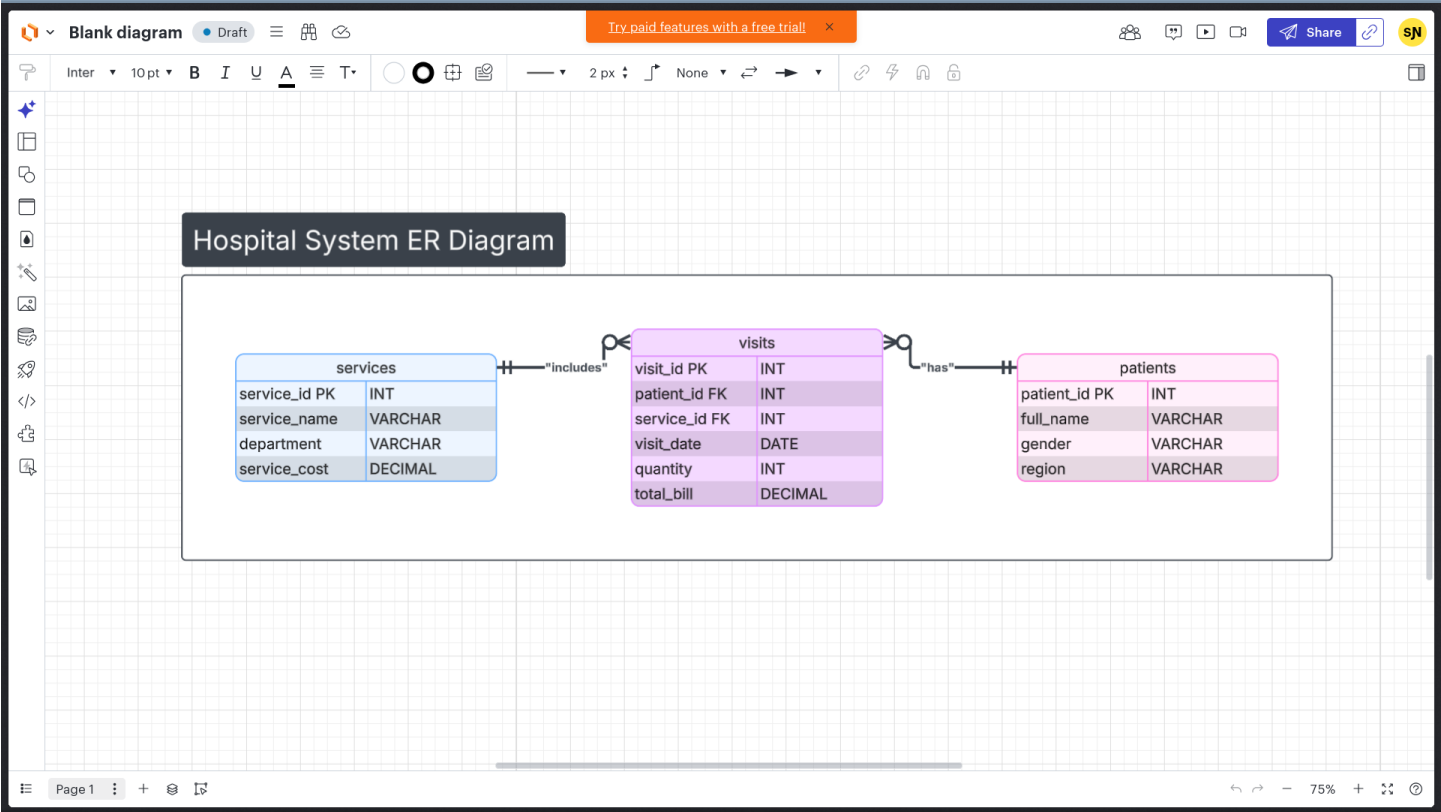
Task completed in 0.328 seconds

Table PATIENTS created.

Table SERVICES created.

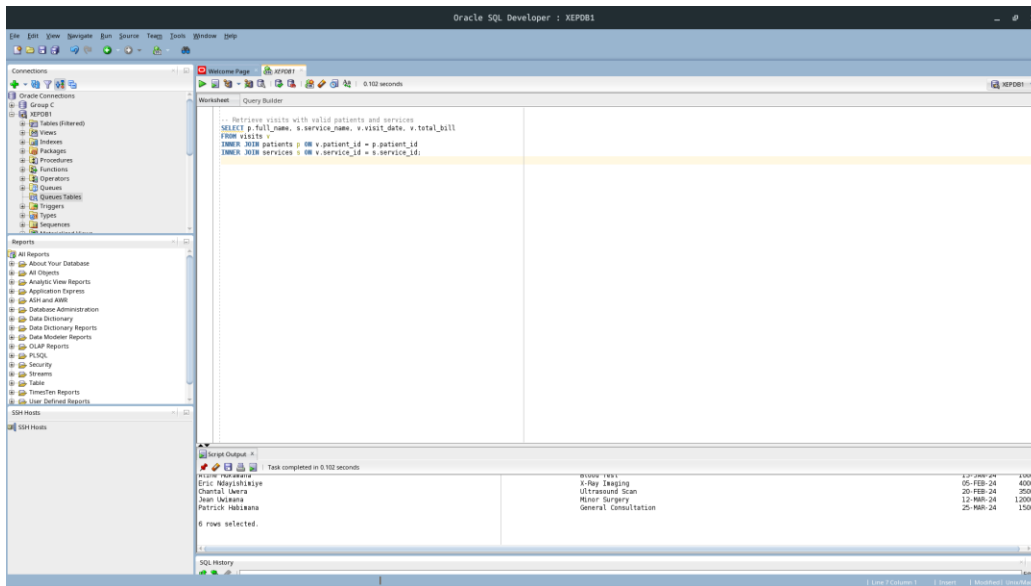
Table VISITS created.

SQL History



# STEP 4: PART A — SQL JOINS Implementation

## INNER JOIN — Valid Visits



The screenshot shows the Oracle SQL Developer interface. The query editor contains the following SQL code:

```
-- Retrieve 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
```

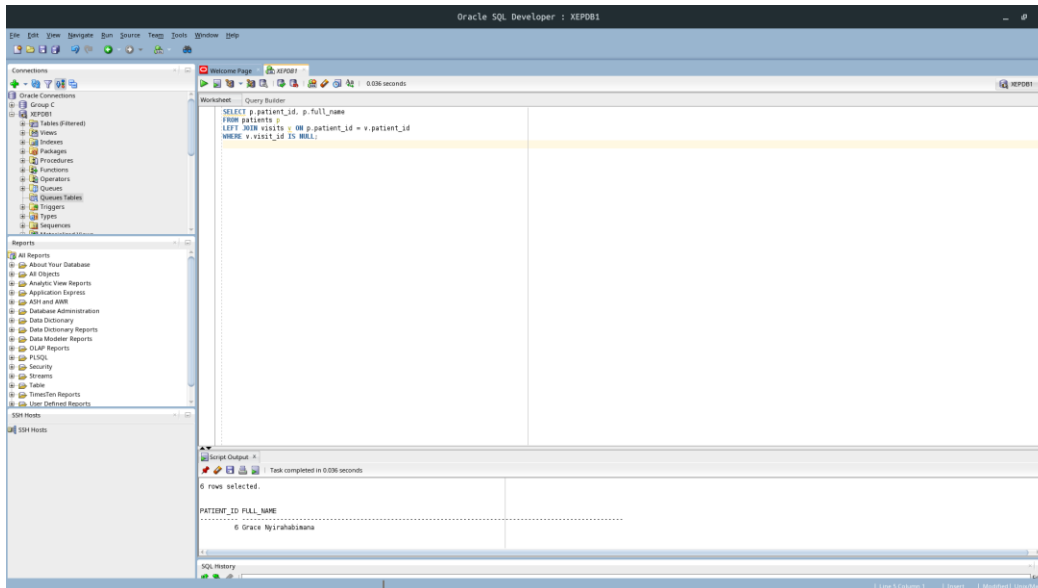
The SQL History pane at the bottom shows the query was executed successfully, returning 6 rows selected. The results are displayed in a table with 5 columns: PATIENT\_ID, FULL\_NAME, SERVICE\_ID, VISIT\_DATE, and TOTAL\_BILL.

PATIENT_ID	FULL_NAME	SERVICE_ID	VISIT_DATE	TOTAL_BILL
1000	Grace Nyirahabimana	1	05-FEB-24	4000
1001	Eric Ndayishimiye	2	20-FEB-24	2500
1002	Charles Muryu	3	12-MAR-24	12000
1003	Jean Umukunda	4	25-MAR-24	1500
1004	Patrick Nshimana	5	05-FEB-24	4000
1005	Grace Nyirahabimana	6	20-FEB-24	2500

## 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. The query editor contains the following SQL code:

```
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;
```

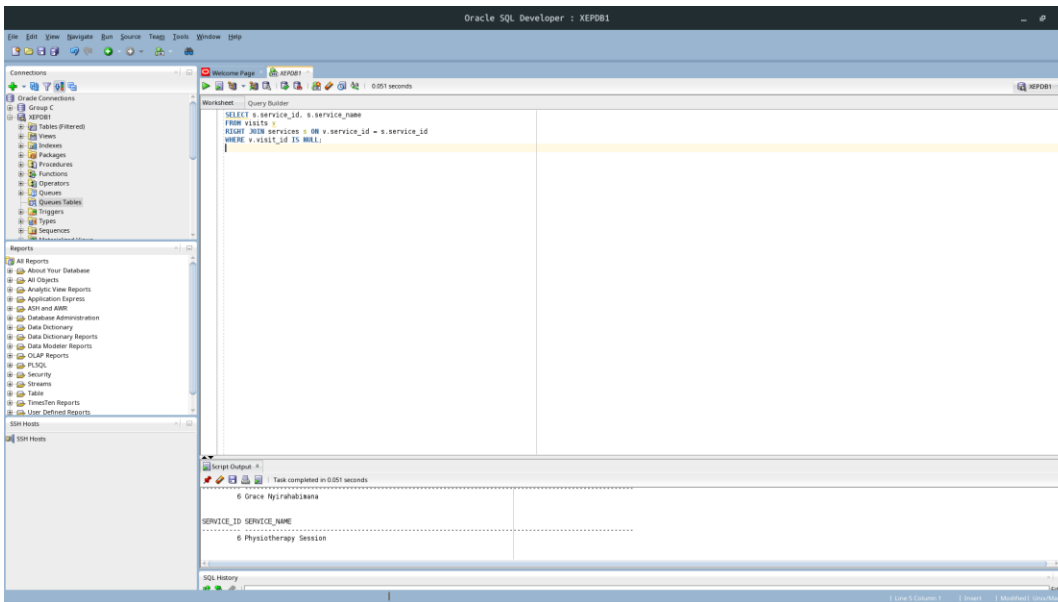
The SQL History pane at the bottom shows the query was executed successfully, returning 0 rows selected. The results are displayed in a table with 2 columns: PATIENT\_ID and FULL\_NAME.

PATIENT_ID	FULL_NAME
------------	-----------

## Interpretation:

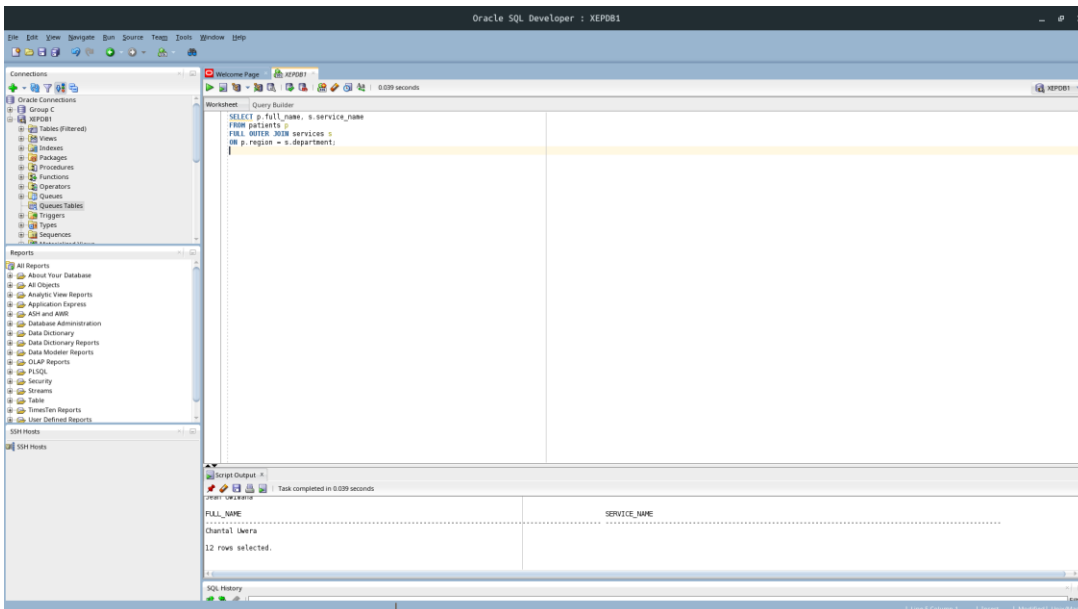
Identifies registered patients who have never visited the hospital.

## RIGHT JOIN — Services Never Used



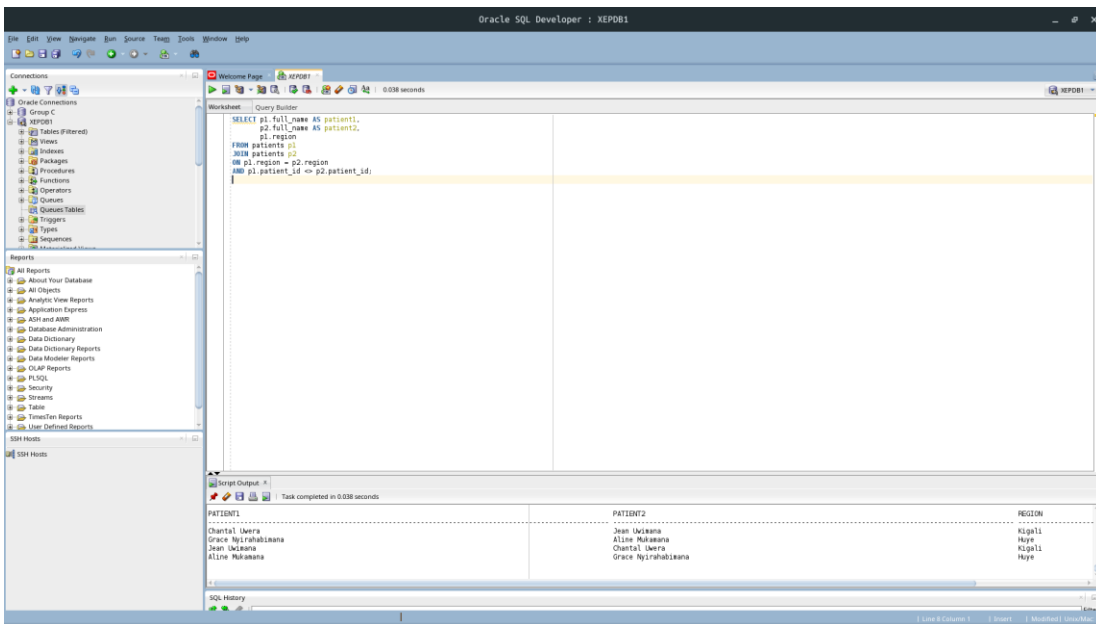
**Interpretation:**  
Highlights medical services that have not been utilized.

## FULL OUTER JOIN — Patients & Services



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

## SELF JOIN — Patients from Same Region

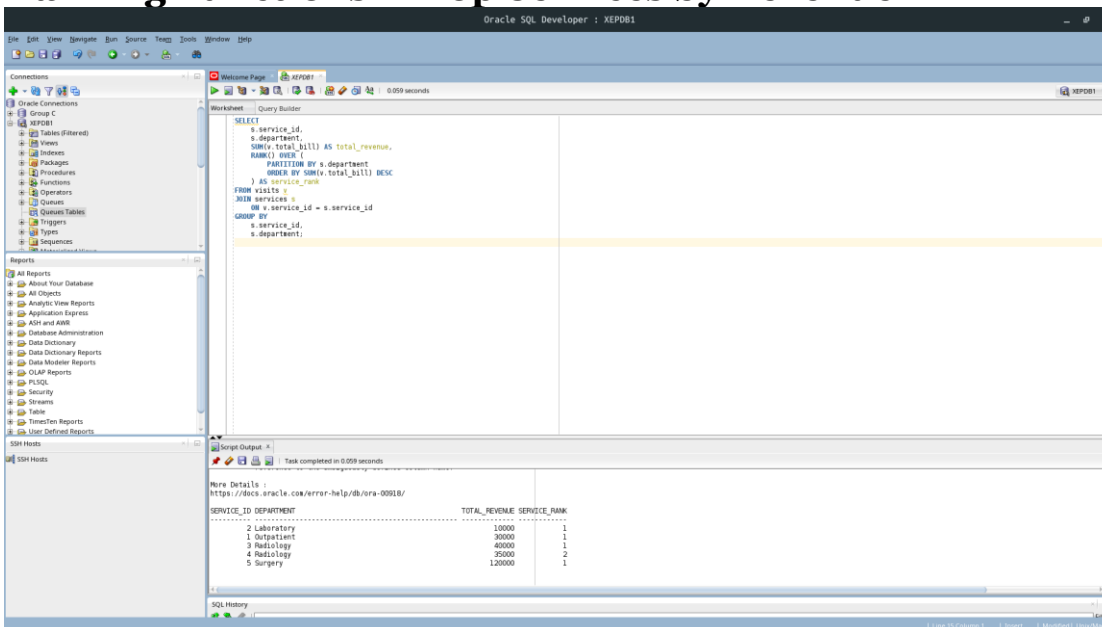


## Interpretation:

Helps analyze patient distribution within the same region.

## STEP 5: PART B — Window Functions

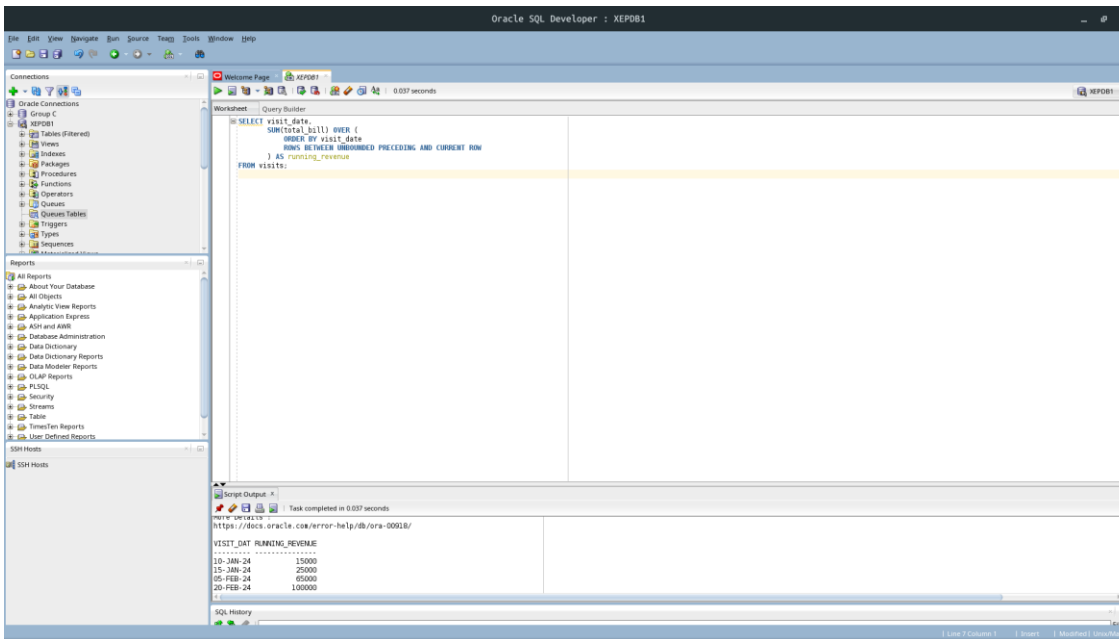
### Ranking Functions — Top Services by Revenue



## Interpretation:

Ranks medical services within each department based on revenue.

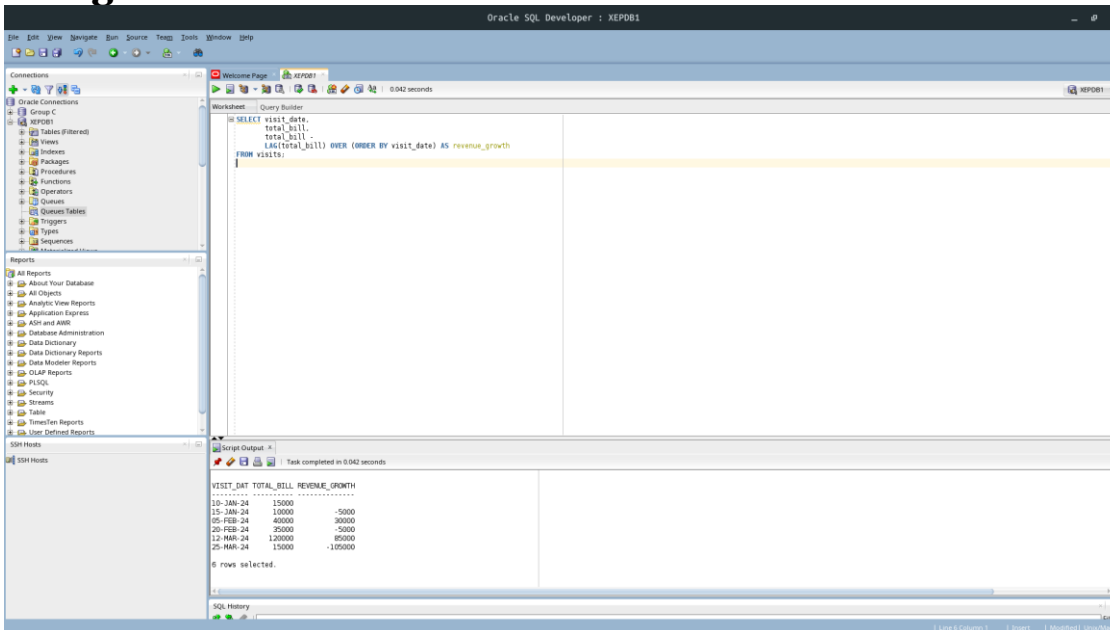
## Aggregate Window — Running Revenue



## Interpretation:

Shows cumulative hospital revenue growth over time.

## Navigation Function — Month-to-Month Growth



## Interpretation:

Tracks changes in revenue between consecutive periods.

## Distribution Functions — Patient Segmentation

Oracle SQL Developer : XEPDB1

Query Builder

```

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

```

Script Output: A

Task completed in 0.033 seconds

6 rows selected.

PATIENT_ID	TOTAL_SPENT	SPENDING_QUARTILE
1	135000	1
3	40000	1
4	25000	2
5	15000	3
2	10000	4

SQL History

## Interpretation:

Segments patients into quartiles based on hospital spending.

## Moving Average (3-Month)

Oracle SQL Developer : XEPDB1

Query Builder

```

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

```

Script Output: A

Task completed in 0.025 seconds

6 rows selected.

VISIT_DAT	THREE_MONTH_AVG
10-JAN-24	15000
15-JAN-24	12500
05-FEB-24	21666.6667
20-FEB-24	28333.3333
12-MAR-24	6000
25-MAR-24	56666.6667

SQL History

## 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.*