P5.PSM Implementation

Team Project Submission
Course: DAMG 6210 Data mgt and Database Design
Northeastern University

Group2 Members:
- Devanshu Chicholikar
- Saurabh Kashyap
- Joseph Alex Chakola
- XingXing Xiao

OUR GitHub URL is below (public repo):
https://github.com/starsbro/DAMG6210_Group02

Requirement:

Database Objects and Features:

- At least **3 stored procedures** with **input and output parameters**, **transaction management**, and **error handling** (e.g., using `TRY...CATCH` blocks).
- At least **3 views**, commonly used for reporting purposes.
- At least **3 user-defined functions** (UDFs).
- At least **1 DML trigger** (e.g., for auditing or enforcing business rules).
- **Column data encryption** for sensitive information.
- At least **3 non-clustered indexes** for performance optimization.
- **Data visualization** using Power BI or Tableau. –We choose Tableau.
- A **graphical user interface (GUI)** for CRUD operations (optional, bonus).

Submission Includes:

- `create_tables.sql` – The DDL script originally submitted for P4, updated with any modifications.
- `insert_script.sql` – The insert script from P4, revised as needed.
- `psm_script.sql` – Contains:
    - All **stored procedures**, with **transaction management and error handling**.
    - **User-defined functions (UDFs)**.
    - **Views**.
    - **Triggers**.
- `indexes_script.sql` – A script defining all non-clustered indexes.
- `encryption_script.sql` – A script implementing column-level encryption.
- `visualization_report.pdf` – A PDF showing your dashboard and analysis.
- **Data Visualization Files** – A zipped folder containing all assets used for the report, dashboard, and optional GUI.

Below is our dashboard example from Tableau. Please review.

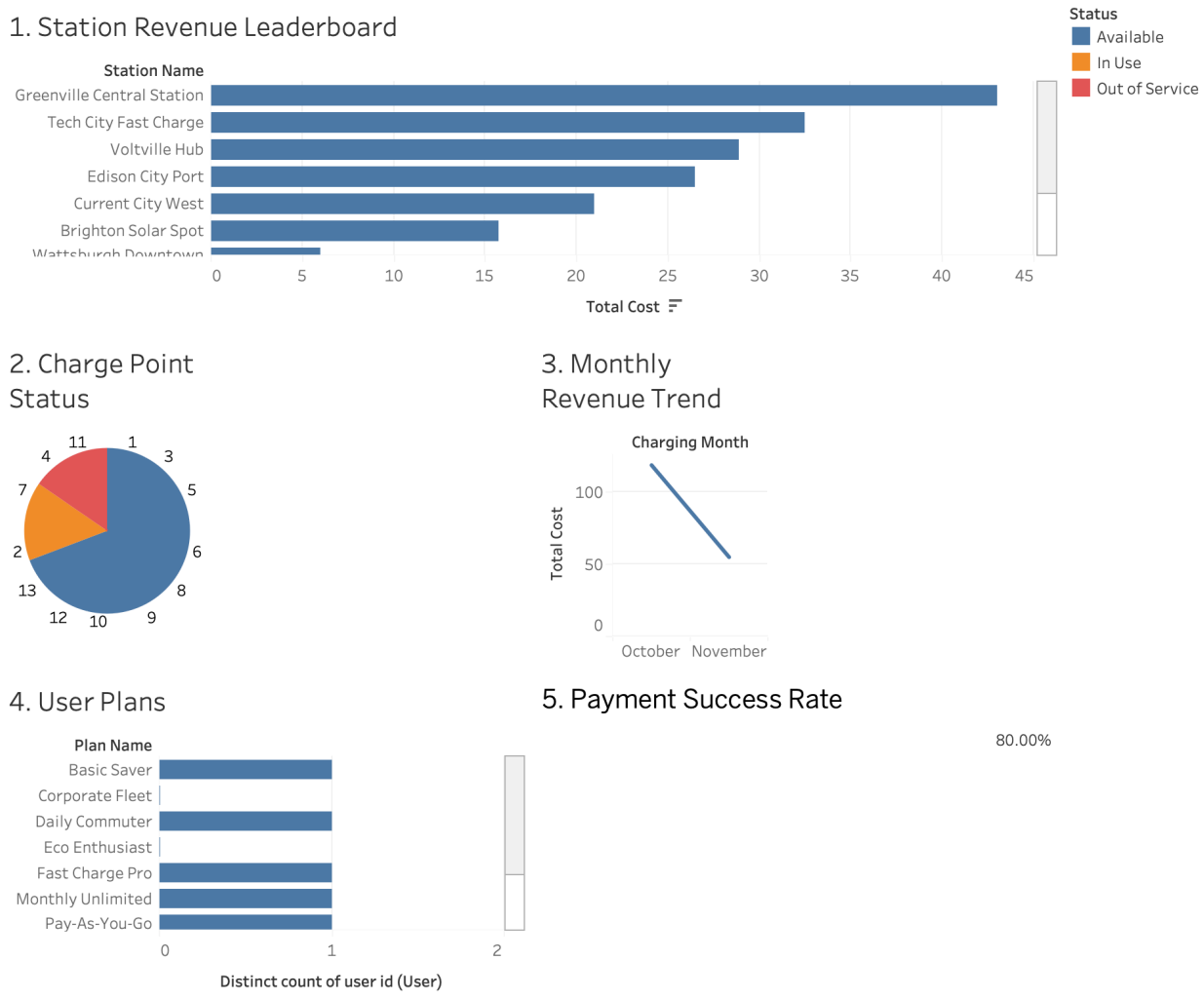

Fig 1 Dashboard of EV Charging System

P5 PSM Implementation Summary (EV Charging System)

The provided SQL scripts demonstrate a complete implementation of **Programmable Stored Modules (PSM)** and database enhancements for the **EV Charging System** project (P5). This implementation covers advanced database features required for operational support, financial reporting, maintenance tracking, and security.

Key Implemented Database Objects

The following objects were created using SQL Server Management Studio (SSMS) commands:

1. Stored Procedures (PSPs)

Three stored procedures were implemented, all adhering to requirements by including **input/output parameters**, **transaction management**, and **error handling** (TRY...CATCH blocks):

- **storedProcedures_ProcessChargingSession**: This is the core transactional procedure. It handles recording a completed charging session, updates the associated charge point status

to 'Available', and automatically generates a corresponding Invoice record. It outputs the new Session_id and Invoice_id.
- **storedProcedures_UpdateChargePointStatus**: A utility procedure to safely change the status of a specific Charge_Point (e.g., from 'In Use' to 'Out of Service'), validating the input status against the table's constraints.
- **storedProcedures_AssignTechnicianToMaintenance**: Manages workflow by assigning a Technician to a Maintenance_Record and updating the record's status to 'In Progress' within a transaction.

2. User-Defined Functions (UDFs)

Three user-defined functions were created to perform repeatable, specific calculations:

- **function_CalculateReservationDurationMinutes**: A scalar function that calculates the time elapsed between the start_time and end_time of a reservation in minutes.
- **function_GetTotalUserEnergyConsumption**: A scalar function that returns the total energy consumed (in kWh) by a specific User across all their historical Charging_Session records.
- **function_GetTechnicianSkills**: A table-valued function that returns a table listing the skill_name and description for a given technician_id. (Note: The attempted function_EncryptCardNumber was blocked by SQL Server due to the use of side-effecting commands.)

3. Views (Reporting)

Three views were created to simplify complex queries often used for reporting and data visualization (e.g., in Tableau):

- **view_MonthlyChargingReport**: A detailed report view that aggregates data by month and station, showing the total number of sessions, total energy (kWh) delivered, and total revenue.
- **view_ActiveUserSubscriptions**: Lists all users who currently have an active subscription (where end_date is null or in the future), providing the plan details and discount rate.
- **view_ChargePointStatusSummary**: Provides a summary of every charge point, including its associated Station Name, city, charger_type, power_rating, and current status.

4. Trigger (Business Rule Enforcement)

One **Data Manipulation Language (DML) trigger** was implemented to enforce a key business rule:

- **trigger_UpdateReservationOnSessionInsert**: This trigger executes **after** a new Charging_Session record is inserted. It automatically finds any corresponding 'Confirmed' Reservation record (based on matching charge point ID and overlapping time window) and updates its status to **'Completed'**.

Security and Performance Enhancements

5. Column-Level Encryption

Sensitive data fields (card_number in Credit_Card and Debit_Card) were encrypted using the **Symmetric Key Encryption** model:

- A **Database Master Key** was created.
- A **Certificate** (CardDataCertificate) was created, secured by the Master Key.
- A **Symmetric Key** (CardDataSymmetricKey) using AES 256 encryption was created, secured by the Certificate.
- Existing data was updated using EncryptByKey() after opening the Symmetric Key.

6. Non-Clustered Indexes

Several non-clustered indexes were defined to optimize query performance on frequently queried columns and foreign keys:

- **Index_PersonAddress_Type**: Improves lookups on the Person_Address table using the address_type.
- **Index_ChargingSession_EndTime**: Optimizes queries that filter or sort by the most recent end_time of charging sessions.
- **Index_Vehicle_ConnectorType**: Speeds up queries that match vehicles to compatible charge points based on the connector_type.