Informe



Prueba Parcial

N°2

Taller Bases de Datos

* Integrantes :
  + FELIX ELIAS VARGAS OLIVARES
  + GISLAINE STEPHANIE PAREDES FIGUEROA
  + MATIAS PADILLA GUTIERREZ
  + THAIS LUCIA CAAMAÑO BUGUEÑO
* Profesor : LEONARDO HERNANDEZ
* Fecha de entrega : 22/10/2025
* **Descripción del proyecto**

El proyecto tiene como propósito optimizar la gestión de arriendo de Rent a House y la administración de clientes dentro de la corredora de propiedades.

Se realizarán desarrollos de diferentes componentes PL/SQL, implementando funciones, procedimientos, packages y triggers que automatizan y analizan distintos procesos del negocio inmobiliario

* **Alcance del proyecto**

Reglas de negocio

**1ra Regla:** El sistema tiene como objetivo gestionar la inserción de registros relacionados con propiedades , propietarios y clientes dentro del sistema inmobiliario.

**Inserción de propiedad** : Al insertar una propiedad, el sistema se encargará de auto generar un id único y cuando cumplen todos los requisitos para la inserción, se agrega y se hace un auto commit.

**Inserción de Propietario**: El sistema hace la inserción y como llave primaria se usará el número de rut, todos los datos son no nullable a excepción de celular y id comuna

**Inserción de cliente**: El sistema hace la inserción y como llave primaria se usará el número de rut, todos los datos son no nullable a excepción de celular

**2da Regla:** La empresa requiere identificar propiedades disponibles para subarriendo. El sistema debe:

* **Calcular el valor de subarriendo** tomando el **80%** del valor del arriendo, sumando los gastos comunes y aplicando un margen de ganancia del **20%**.
* **Procesar todas las propiedades registradas**, verificando si cumplen dos condiciones:
  1. Superficie mínima de **50 m²**.
  2. No estar arrendadas (comparando con la tabla de propiedades arrendadas).
* **Registrar automáticamente** las propiedades que cumplen estos criterios, con los siguientes datos(nro,propiedad,dirección,superficie,nro dormitorios,nro baños,valor arriendo,valor gastos comunes,valor subarriendo,rut del propietario y del rut del empleado junto con su valor de subarriendo, en la tabla PROP\_POTENCIAL\_ARRIENDO.

Esta regla permite a la empresa detectar rápidamente propiedades rentables, optimizando la toma de decisiones para el subarriendo.

**3ra Regla:**

El sistema debe permitir obtener la información de los clientes que han arrendado propiedades, filtrando los resultados según el estado civil (casado, soltero, separado o viudo). Cada vez que un cliente realiza un arriendo, el sistema se actualiza y al generar el reporte automáticamente muestra los datos personales del cliente junto con el tipo de propiedad que arriendo y la cantidad del tipo de propiedad que este tiene. Con esto se puede analizar el comportamiento de los clientes al arrendar, la preferencia que tienen por las propiedades según su estado civil. Permite a la empresa tomar decisiones estratégicas y ofrecer una mejor experiencia al cliente.

**Se hará lo siguiente:**

* **Registro de información:** Inserta automáticamente propiedades, clientes y propietarios en la base de datos, asegurando que los datos principales estén disponibles para el sistema.
* **Cálculo y carga de propiedades potenciales:** Identifica propiedades disponibles para subarriendo, calcula su valor estimado y las guarda en una tabla específica para análisis.
* **Generación de reportes automáticos:** Produce reportes sobre clientes y propiedades arrendadas, filtrando información por estado civil y tipo de propiedad, y actualiza los reportes en tiempo real mediante un trigger.
* **Front desde la vista administrador** para hacer inserciones,cálculos y reportes

**No se hará lo siguiente:**

* FRONT DESDE LA VISTA DE USUARIO
* **Tecnologías Utilizadas**

1. Oracle SQL developer para almacenar y gestionar la información en una base de datos funcional
2. SQL y PL/SQL servirán para procedimientos almacenados y las secuencias de datos y manejo de excepciones
3. Git y Github para almacenar, gestionar y unir los scripts de SQL
4. Para el FRONT : REACT conectado a oracle

* **Funciones**

Bloques de código que son reutilizables que devuelven un valor único después de realizar una operación específica. Son similares a las funciones matemáticas: reciben parámetros de entrada y producen un resultado de salida.

Entres sus características :

* **Return:** siempre se debe usar para devolver un único valor del tipo especificado.
* **Parámetros:**puede tener parámetros de entrada(IN) ,de salida (OUT) o ambos.Por defecto son de entrada.
* **Desventajas de las funciones**
* **Sentencias DML :** no puede usarse sentencias DML como ( INSERT,UPDATE Y DELETE),si se usa en consultas SQL a menos que esté en un bloque autónomo.
* **Impacto en el rendimiento :** si se llaman muchas veces las funciones dentro de una consulta grande puede hacerla más lenta.
* **Dificultad para depurar :** encontrar errores dentro de las funciones llamadas desde SQL puede ser más difícil que al ejecutar procedimientos directamente.
* **Mayor complejidad para entender:** si se desarrollan muchas funciones pequeñas,el sistema se vuelve más difícil de comprender y mantener
* **Ventajas de las funciones**
* **Reutilización:** se puede reutilizar código ya que las funciones se pueden usar en distintas partes como por ejemplo llamar a la función dentro de procedimientos sin tener que escribirlo nuevamente.
* **Organización del código:** facilita dividir un programa grande en partes más pequeñas.
* **Mantenimiento :** facilita el mantenimiento,visualizandose en el momento que si hay cambios en la lógica ,solo se modifica la función y no donde se usó.
* **Procedimientos almacenados**

Los procedimientos almacenados son bloques de código que se guardan en la base de datos para posteriormente ser ejecutados en el servidor en cuanto se invoca, sus características clave son el precompilado, que son reutilizables, es decir, se pueden invocar o llamar múltiples veces según sea necesario, centralizados puesto que el código reside en el servidor de la base de datos y aceptan parámetros de entrada.

* **Desventajas de los procedimientos almacenados**

Dependencia del DBMS (Sistema de Gestión de Bases de Datos), es decir, no son portables entre distintas bases de datos, las herramientas de debug son más limitadas y generalmente más difíciles de ocupar, si no se gestiona de manera adecuada se puede convertir en un cuello de botella si consumen muchos recursos de la base de datos.

* **Ventajas de los procedimientos almacenados**

Las ventajas que ofrece almacenar los procedimientos son mejor rendimiento al ser scripts precompilados, mayor seguridad por la abstracción y control de accesos a las tablas, optimiza el mantenimiento ya que se encuentra con lógica centralizada la cual es más fácil de modificar y mejora la integridad.

* **El propósito de los procedimientos y funciones en el contexto del proyecto**

Los procedimientos y funciones están diseñados para automatizar y optimizar la gestión de datos en el sistema inmobiliario Rent a House, facilitando el registro, análisis y manejo de propiedades, clientes y reportes, y agilizando las operaciones internas de la empresa.

* **Utilización de los procedimientos y funciones en el contexto del proyecto**
* **PKG FELIX:** agrupa procedimientos para **registrar información básica**, como propiedades, clientes y propietarios. Su objetivo es mantener actualizada la base de datos de Rent a House y asegurar la correcta inserción de los datos principales del sistema.
* **PKG THAIS:** incluye una función que **calcula el valor de subarriendo** de las propiedades y un procedimiento que **carga las propiedades potenciales** para arriendo. Este paquete apoya la toma de decisiones comerciales y la gestión de propiedades disponibles.
* **PKG GISLAINE:** contiene una función y un procedimiento orientados a **generar reportes estadísticos** sobre los clientes según su estado civil. Permite analizar el perfil de los arrendatarios y mantener la información de reportes actualizada mediante un **trigger automático**.

(se adjunta código al final para visualizar)

* **Packages**

Agrupaciones lógicas que organizan procedimientos almacenados, funciones, variables y otros objetos relacionados en un solo contendor, contienen un header que consiste en la parte visible (interfaz pública) y un body que contiene toda la lógica interna, permiten encapsular funcionalidades complejas y permite la gestión de permisos al mismo, ya sea gradual, completa o nula

* **Desventajas al momento de utilizar de los triggers**

Pueden dificultar la depuración, al utilizarlos mucho pueden generar conflictos o errores en cascada lo que puede terminar en múltiples errores de compilación.

* **Ventajas al momento de utilizar de los triggers**

Se ejecutan automáticamente al momento de ocurrir un evento especificado, permite automatizar procesos o en su defecto prevenir errores o eventos no deseados, sus usos pueden variar según lo que se intente prever.

* **Cómo los packages ayudan a organizar el código en el proyecto**

Los packages **PKG\_FELIX, PKG\_THAIS y PKG\_GISLAINE** organizan el código del proyecto Rent a House de forma modular y eficiente:

* **Modularidad**: Dividen la lógica en funciones específicas:
  + **PKG FELIX:** Registro de propiedades, clientes y propietarios.
  + **PKG THAIS:** Cálculo y carga de propiedades para subarriendo.
  + **PKG GISLAINE:** Generación de reportes por estado civil.
* **Reusabilidad**: Centralizan funciones como calcular subarriendo y **fn\_porcentaje\_estcivil,** evitando duplicación.
* **Mantenimiento**: Facilitan cambios al localizar la lógica en un solo lugar.
* **Legibilidad**: Nombres claros y estructura separada (especificación/cuerpo) mejoran la comprensión.
* **Escalabilidad**: Permiten agregar nuevas funcionalidades sin reestructurar el sistema.

**Ejemplo**: PKG FELIX registra una propiedad, PKG THAIS la evalúa para subarriendo, y PKG GISLAINE genera reportes automáticos tras un arriendo. Esto asegura un código organizado, mantenible y escalable.

* **Relación entre los packages con los procedimientos y funciones**

Los packages en bases de datos (como en Oracle PL/SQL) son estructuras que organizan y agrupan procedimientos y funciones relacionados, proporcionando una forma modular de gestionar código de la siguiente manera:

* **Estructura Modular**: Los packages tienen una **especificación** (define nombres y parámetros de procedimientos/funciones) y un **cuerpo** (contiene la lógica). Esto separa la interfaz pública de la implementación, mejorando la organización.
* **Encapsulamiento**: Agrupan procedimientos y funciones con propósitos similares, como operaciones de registro, cálculos o reportes, manteniendo el código ordenado y accesible.
* **Interdependencia**: Los procedimientos y funciones dentro de un package pueden llamarse entre sí, reutilizando lógica. Por ejemplo, un procedimiento puede usar una función del mismo package para cálculos internos.
* **Reusabilidad y Acceso**: Los elementos del package se invocan con la sintaxis PACKAGE.NOMBRE (ejemplo: PACKAGE.calcular valor), permitiendo su uso desde otros programas, scripts o triggers.
* **Mantenimiento**: Al centralizar la lógica relacionada, los packages facilitan la actualización y depuración del código, ya que los cambios se realizan en un solo lugar.
* **Cómo se utilizaron los packages en el contexto del proyecto** 
  + El **package PKG\_FELIX:** se utilizó para **gestionar la inserción de datos en la base**, como el registro de propiedades, clientes y propietarios. Esto permitió realizar las cargas iniciales de información de manera controlada y evitar errores al insertar datos en múltiples tablas relacionadas.
  + El **package PKG\_THAIS** :se aplicó en la **identificación de propiedades potenciales para subarriendo**, calculando automáticamente los valores estimados mediante una función interna. Gracias a este package, se automatizó el proceso de análisis de propiedades disponibles y se almacenaron los resultados en una tabla específica.
  + El **package PKG\_GISLAINE:** se empleó para la **generación de reportes estadísticos**, especialmente sobre el estado civil de los clientes. Además, se integró con un *trigger* que ejecuta el procedimiento del paquete cada vez que se registra un nuevo arriendo, asegurando que los reportes se mantengan actualizados en tiempo real.

En resumen, los packages actúan como contenedores que estructuran, encapsulan y conectan procedimientos y funciones, promoviendo modularidad, reutilización y facilidad de mantenimiento en sistemas de bases de datos.

* **Triggers**

Los triggers o gatillos son bloques de código que están diseñados para activar líneas de código de manera automática, permiten los INSERT, UPDATE y DELETE, sus usos pueden ser temporales de manera que tienen las palabras reservadas BEFORE y AFTER, garantizan acciones y consistencia frente a posibles errores o cambios de datos.

* **Desventajas en la utilización de triggers.**

Se ejecutan acciones automáticas que no son evidentes, al ser ejecutadas en segundo plano complican el troubleshooting, su lógica es más difícil de rastrear en relación a reglas empresariales ya que puede causar dispersión, lo cuál es más complejo de rastrear.

* **Ventajas en la utilización de triggers**

Permiten validaciones más complejas que las mismas llaves foráneas, su función permite optimizar procesos de manera automática frente a ciertos eventos o condiciones, permiten cálculos en tiempo real asegurando datos más certeros y en reglas de seguridad ofrecen una capa más compleja.

* **El propósito de los triggers en el contexto del proyecto**

En el contexto el propósito del trigger es automatizar los procesos dentro de la Base de datos de Rent a House sin requerir una intervención manual. La función principal es ejecutar acciones específicas de forma automática cada vez que ocurre un evento como inserción, actualización o eliminación de los datos

* **Cómo se utilizaron los triggers en el contexto del proyecto**

Se implementó el trigger **trg\_auto\_reporte\_estcivil** el cual mantiene sincronizada la información entre la BD y el front, cada vez que un cliente realiza un arriendo y se inserta un registro en la propiedad arrendada, el trigger se ejecuta automáticamente. Es decir que si uno por filtro busca estado civil (soltero, viudo, casado, separado) el trigger asegura que la información está procesada y almacenada en la tabla **reporte\_cli\_estcivil**, automatizando la actualización de los datos analíticos.

* **Resumen**

El proyecto de RENT A HOUSE busca mejorar la gestión de arriendos de propiedades (casas, departamentos, locales, etc.) con un sistema informático que automatiza procesos clave. El código PL/SQL maneja:

1. Inserción de propiedades: Registra nuevas propiedades con IDs automáticos y detalles como dirección, superficie o valor de arriendo.
2. Propiedades para subarriendo: Usar una función y un procedimiento dentro de un package para identificar propiedades disponibles (superficie ≥ 50 m²), aplica un descuento del 20% y calcula un precio de subarriendo con 20% de margen.
3. Generar reportes : permite obtener y actualizar automáticamente la información de los clientes que han arrendado propiedades, filtrando los datos según su estado civil. Esto facilita analizar el comportamiento y las preferencias de los clientes, apoyando la toma de decisiones estratégicas de la empresa.

Además de un front para visualizar el comportamiento de dicho código

* **Impacto del proyecto**
  + **Automatización de procesos:** Permite registrar propiedades, clientes y propietarios de manera rápida y confiable, reduciendo errores manuales.
  + **Gestión de subarriendo:** Identifica automáticamente las propiedades disponibles y calcula su valor estimado, facilitando decisiones sobre rentabilidad.
  + **Generación de reportes:** Proporciona información actualizada sobre clientes y propiedades, incluyendo análisis por estado civil y tipo de propiedad.
  + **Optimización de decisiones estratégicas:** Los datos procesados permiten a la empresa planificar acciones comerciales y mejorar la oferta de propiedades.
  + **Mejora de la experiencia del cliente:** La información precisa y actualizada permite ofrecer un servicio más eficiente y personalizado.
* **Recomendaciones**
* Validación de la entrada de datos: Incluir controles adicionales en los procedimientos para comprobar que los datos que se están insertando (RUT, valores de arriendo, superficie, etc.) corresponden a datos válidos y correctos antes de ser insertadas en la base de datos correspondiente.
* Optimización de las consultas: Examinar los cursores y las consultas a efectos de mejorar el rendimiento del sistema, sobre todo cuando hay grandes volúmenes de propiedades o de clientes.
* Manejo de errores: Ampliar las excepciones y los registros de log en los paquetes para detectar y registrar errores durante la ejecución de procedimientos o funciones de los paquetes.
* Actualización periódica de la tabla de propiedades potenciales: Tratar de no truncar la tabla dentro del ciclo; en su lugar, se podrían limpiar los registros antes de iniciar la inserción para mejorar aún más el rendimiento.
* Documentación y mantenimiento: Mantener una documentación actualizada y clara de cada uno de los procedimientos, funciones y triggers para favorecer modificaciones futuras o para ampliar el sistema si se requiere.
* Extensión de reportes: Incluir nuevos filtros y nuevas métricas en los reportes, como por ejemplo rangos de edad, tipo de propiedades o ubicación, para permitir disponer de análisis del comportamiento de las propiedades más completos.
* **Anexos**

**Codigo:**

CREATE TABLE REPORTE\_CLI\_ESTCIVIL (

ID\_REPORTE NUMBER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,

DESC\_ESTCIVIL VARCHAR2(30),

PORCENTAJE NUMBER(5,2),

RUT\_CLIENTE VARCHAR2(11),

NOMBRE\_COMPLETO VARCHAR2(120),

DESC\_TIPO\_PROPIEDAD VARCHAR2(60),

CANTIDAD\_PROPIEDAD NUMBER

);

CREATE TABLE PROP\_POTENCIAL\_ARRIENDO (

NRO\_PROPIEDAD NUMBER(6,0),

DIRECCION\_PROPIEDAD VARCHAR2(60),

SUPERFICIE NUMBER(8,2),

NRO\_DORMITORIOS NUMBER(1,0),

NRO\_BANOS NUMBER(1,0),

VALOR\_ARRIENDO NUMBER(7,0),

VALOR\_GASTO\_COMUN NUMBER(6,0),

PRECIO\_SUBARRIENDO NUMBER(7,0),

NUMRUT\_PROP NUMBER(10,0),

NUMRUT\_EMP NUMBER(10,0),

FECHA\_REGISTRO DATE DEFAULT SYSDATE);

CREATE OR REPLACE PACKAGE PKG\_FELIX

IS

PROCEDURE PS\_INSERCION\_DE\_PROPIEDAD (

ENTREGA\_PROPIEDAD DATE,

DIRECCION VARCHAR2,

SUPERFICIE NUMBER,

N\_DORMITORIOS NUMBER,

N\_BANIOS NUMBER,

VALOR\_ARRIENDO NUMBER,

VALOR\_GASTO NUMBER,

NUM\_RUT\_PROPIETARIO NUMBER,

ID\_COMUNA NUMBER,

ID\_TIPO\_PROPIEDAD VARCHAR2,

NUMRUT\_EMP NUMBER

);

PROCEDURE PS\_INSERCION\_DE\_CLIENTE (NUMRUT\_CLI NUMBER,

DVRUT\_CLI NUMBER,

APPATERNO\_CLI VARCHAR2,

APMATERNO\_CLI VARCHAR2,

NOMBRE\_CLI VARCHAR2,

DIRECCION\_CLI VARCHAR2,

ID\_ESTCIVIL NUMBER,

FONOFIJO\_CLI NUMBER,

CELULAR\_CLI NUMBER,

RENTA\_CLI NUMBER);

PROCEDURE PS\_INSERCION\_DE\_PROPIETARIO (NUMRUT\_PROP NUMBER,

DVRUT\_PROP NUMBER,

APPATERNO\_PROP VARCHAR2,

APMATERNO\_PROP VARCHAR2,

NOMBRE\_PROP VARCHAR2,

DIRECCION\_PROP VARCHAR2,

ID\_ESTCIVIL NUMBER,

FONOFIJO\_PROP NUMBER,

CELULAR\_PROP NUMBER,

ID\_COMUNA NUMBER

);

END PKG\_FELIX;

CREATE OR REPLACE PACKAGE BODY PKG\_FELIX

IS

PROCEDURE PS\_INSERCION\_DE\_PROPIEDAD(

ENTREGA\_PROPIEDAD DATE,

DIRECCION VARCHAR2,

SUPERFICIE NUMBER,

N\_DORMITORIOS NUMBER,

N\_BANIOS NUMBER,

VALOR\_ARRIENDO NUMBER,

VALOR\_GASTO NUMBER,

NUM\_RUT\_PROPIETARIO NUMBER,

ID\_COMUNA NUMBER,

ID\_TIPO\_PROPIEDAD VARCHAR2,

NUMRUT\_EMP NUMBER

)

IS

V\_MAYOR\_ID NUMBER(10);

BEGIN

SELECT NVL(MAX(NRO\_PROPIEDAD), 0) + 1

INTO V\_MAYOR\_ID

FROM PROPIEDAD;

INSERT INTO PROPIEDAD (

NRO\_PROPIEDAD,

FECHA\_ENTREGA\_PROPIEDAD,

DIRECCION\_PROPIEDAD,

SUPERFICIE,

NRO\_DORMITORIOS,

NRO\_BANOS,

VALOR\_ARRIENDO,

VALOR\_GASTO\_COMUN,

NUMRUT\_PROP,

ID\_COMUNA,

ID\_TIPO\_PROPIEDAD,

NUMRUT\_EMP

)

VALUES (

V\_MAYOR\_ID,

ENTREGA\_PROPIEDAD,

DIRECCION,

SUPERFICIE,

N\_DORMITORIOS,

N\_BANIOS,

VALOR\_ARRIENDO,

VALOR\_GASTO,

NUM\_RUT\_PROPIETARIO,

ID\_COMUNA,

ID\_TIPO\_PROPIEDAD,

NUMRUT\_EMP

);

COMMIT;

END PS\_INSERCION\_DE\_PROPIEDAD;

PROCEDURE PS\_INSERCION\_DE\_CLIENTE (NUMRUT\_CLI NUMBER,

DVRUT\_CLI NUMBER,

APPATERNO\_CLI VARCHAR2,

APMATERNO\_CLI VARCHAR2,

NOMBRE\_CLI VARCHAR2,

DIRECCION\_CLI VARCHAR2,

ID\_ESTCIVIL NUMBER,

FONOFIJO\_CLI NUMBER,

CELULAR\_CLI NUMBER,

RENTA\_CLI NUMBER)

IS

BEGIN

INSERT INTO CLIENTE VALUES (NUMRUT\_CLI,

DVRUT\_CLI ,

APPATERNO\_CLI ,

APMATERNO\_CLI ,

NOMBRE\_CLI ,

DIRECCION\_CLI ,

ID\_ESTCIVIL ,

FONOFIJO\_CLI ,

CELULAR\_CLI ,

RENTA\_CLI );

COMMIT;

END PS\_INSERCION\_DE\_CLIENTE;

PROCEDURE PS\_INSERCION\_DE\_PROPIETARIO (NUMRUT\_PROP NUMBER,

DVRUT\_PROP NUMBER,

APPATERNO\_PROP VARCHAR2,

APMATERNO\_PROP VARCHAR2,

NOMBRE\_PROP VARCHAR2,

DIRECCION\_PROP VARCHAR2,

ID\_ESTCIVIL NUMBER,

FONOFIJO\_PROP NUMBER,

CELULAR\_PROP NUMBER,

ID\_COMUNA NUMBER

)

IS

BEGIN

INSERT INTO PROPIETARIO

VALUES

(NUMRUT\_PROP ,

DVRUT\_PROP ,

APPATERNO\_PROP ,

APMATERNO\_PROP ,

NOMBRE\_PROP ,

DIRECCION\_PROP ,

ID\_ESTCIVIL ,

FONOFIJO\_PROP ,

CELULAR\_PROP ,

ID\_COMUNA );

COMMIT;

END PS\_INSERCION\_DE\_PROPIETARIO;

END PKG\_FELIX;

CREATE OR REPLACE PACKAGE PKG\_THAIS

IS

FUNCTION calcular\_subarriendo(p\_valor\_arriendo NUMBER,p\_valor\_gasto\_comun NUMBER) RETURN NUMBER;

PROCEDURE cargar\_propiedades\_potenciales;

END PKG\_THAIS;

CREATE OR REPLACE PACKAGE BODY PKG\_THAIS

IS

FUNCTION calcular\_subarriendo(p\_valor\_arriendo NUMBER,p\_valor\_gasto\_comun NUMBER)

RETURN NUMBER

IS

v\_margen NUMBER := 0.2;

BEGIN

RETURN(((p\_valor\_arriendo \* 0.8)+ p\_valor\_gasto\_comun)\*(1+v\_margen));

END calcular\_subarriendo;

PROCEDURE cargar\_propiedades\_potenciales IS

CURSOR cur\_propiedades IS

SELECT NRO\_PROPIEDAD npropiedad,

FECHA\_ENTREGA\_PROPIEDAD fepropiedad,

DIRECCION\_PROPIEDAD direct,

SUPERFICIE superf,

NRO\_DORMITORIOS ndormitorios,

NRO\_BANOS nbanos,

VALOR\_ARRIENDO varriendo,

NVL(VALOR\_GASTO\_COMUN, 0) vgcomun,

ID\_TIPO\_PROPIEDAD tipopropiedad,

ID\_COMUNA comuna,

NUMRUT\_PROP rutpropietario,

NUMRUT\_EMP rutemp

FROM PROPIEDAD;

BEGIN

FOR reg\_propiedades IN cur\_propiedades LOOP

DECLARE

v\_disponible BOOLEAN := TRUE;

v\_precio\_subarriendo NUMBER;

v\_valor\_arriendo NUMBER;

v\_valor\_gasto\_comun NUMBER;

BEGIN

EXECUTE IMMEDIATE 'TRUNCATE TABLE PROP\_POTENCIAL\_ARRIENDO';

v\_valor\_arriendo := reg\_propiedades.varriendo;

v\_valor\_gasto\_comun := reg\_propiedades.vgcomun;

FOR arrendado IN (

SELECT \*

FROM PROPIEDAD\_ARRENDADA

WHERE NRO\_PROPIEDAD = reg\_propiedades.npropiedad

) LOOP

IF arrendado.FECTER\_ARRIENDO IS NULL

OR SYSDATE BETWEEN arrendado.FECINI\_ARRIENDO AND arrendado.FECTER\_ARRIENDO THEN

v\_disponible := FALSE;

EXIT;

END IF;

END LOOP;

IF reg\_propiedades.superf >= 50 AND v\_disponible THEN

v\_precio\_subarriendo := calcular\_subarriendo(v\_valor\_arriendo, v\_valor\_gasto\_comun);

INSERT INTO PROP\_POTENCIAL\_ARRIENDO (

NRO\_PROPIEDAD, DIRECCION\_PROPIEDAD, SUPERFICIE, NRO\_DORMITORIOS, NRO\_BANOS,

VALOR\_ARRIENDO, VALOR\_GASTO\_COMUN, PRECIO\_SUBARRIENDO, NUMRUT\_PROP, NUMRUT\_EMP

) VALUES (

reg\_propiedades.npropiedad, reg\_propiedades.direct, reg\_propiedades.superf,

reg\_propiedades.ndormitorios, reg\_propiedades.nbanos,

v\_valor\_arriendo, v\_valor\_gasto\_comun, v\_precio\_subarriendo,

reg\_propiedades.rutpropietario, reg\_propiedades.rutemp

);

END IF;

END;

END LOOP;

COMMIT;

END cargar\_propiedades\_potenciales;

END PKG\_THAIS;

CREATE OR REPLACE PACKAGE PKG\_GISLAINE

IS

FUNCTION fn\_porcentaje\_estcivil(p\_desc\_estcivil VARCHAR2) RETURN NUMBER;

PROCEDURE sp\_generar\_reporte\_estcivil(p\_desc\_estcivil VARCHAR2);

END PKG\_GISLAINE;

CREATE OR REPLACE PACKAGE BODY PKG\_GISLAINE

IS

FUNCTION fn\_porcentaje\_estcivil(p\_desc\_estcivil VARCHAR2)

RETURN NUMBER

IS

v\_total NUMBER;

v\_estado NUMBER;

v\_porc NUMBER;

BEGIN

SELECT COUNT(DISTINCT numrut\_cli)

INTO v\_total

FROM propiedad\_arrendada;

SELECT COUNT(DISTINCT pa.numrut\_cli)

INTO v\_estado

FROM propiedad\_arrendada pa

WHERE pa.numrut\_cli IN (

SELECT numrut\_cli

FROM cliente

WHERE id\_estcivil = (

SELECT id\_estcivil

FROM estado\_civil

WHERE UPPER(desc\_estcivil) = UPPER(p\_desc\_estcivil)

)

);

IF NVL(v\_total,0) = 0 THEN

RETURN 0;

ELSE

v\_porc := ROUND((v\_estado / v\_total) \* 100,2);

RETURN v\_porc;

END IF;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RETURN 0;

END fn\_porcentaje\_estcivil;

PROCEDURE sp\_generar\_reporte\_estcivil(p\_desc\_estcivil VARCHAR2)

IS

v\_porcentaje NUMBER;

BEGIN

v\_porcentaje := fn\_porcentaje\_estcivil(p\_desc\_estcivil);

EXECUTE IMMEDIATE 'TRUNCATE TABLE REPORTE\_CLI\_ESTCIVIL';

INSERT INTO REPORTE\_CLI\_ESTCIVIL(

DESC\_ESTCIVIL,

PORCENTAJE,

RUT\_CLIENTE,

NOMBRE\_COMPLETO,

DESC\_TIPO\_PROPIEDAD,

CANTIDAD\_PROPIEDAD

)

SELECT

p\_desc\_estcivil,

v\_porcentaje,

c.numrut\_cli || '-' || c.dvrut\_cli,

c.nombre\_cli || ' ' || c.appaterno\_cli || ' ' || c.apmaterno\_cli,

tp.desc\_tipo\_propiedad,

COUNT(\*) AS cantidad\_propiedad

FROM cliente c

JOIN propiedad\_arrendada pa ON pa.numrut\_cli = c.numrut\_cli

JOIN propiedad p ON p.nro\_propiedad = pa.nro\_propiedad

JOIN tipo\_propiedad tp ON tp.id\_tipo\_propiedad = p.id\_tipo\_propiedad

WHERE c.id\_estcivil = (

SELECT id\_estcivil

FROM estado\_civil

WHERE UPPER(desc\_estcivil)=UPPER(p\_desc\_estcivil)

)

GROUP BY

c.numrut\_cli, c.dvrut\_cli,

c.nombre\_cli, c.appaterno\_cli, c.apmaterno\_cli,

tp.desc\_tipo\_propiedad;

COMMIT;

END sp\_generar\_reporte\_estcivil;

END PKG\_GISLAINE;

CREATE OR REPLACE TRIGGER trg\_auto\_reporte\_estcivil

AFTER INSERT ON propiedad\_arrendada

FOR EACH ROW

DECLARE

v\_desc\_estcivil VARCHAR2(30);

BEGIN

SELECT ec.desc\_estcivil

INTO v\_desc\_estcivil

FROM cliente c

JOIN estado\_civil ec ON ec.id\_estcivil = c.id\_estcivil

WHERE c.numrut\_cli = :NEW.numrut\_cli;

PKG\_GISLAINE.sp\_generar\_reporte\_estcivil(v\_desc\_estcivil);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

NULL;

END trg\_auto\_reporte\_estcivil;