

Lucca Giovane Gome

**ATIVIDADE 1: VIEWS, CTE, WINDOW FUNCTIONS E DESENVOLVIMENTO EM PL/SQL**

**LONDRINA - PR**

**2024**

**Relatório**

## **1. Criação do Banco de Dados**

### **1.1. Scripts de Criação das Tabelas**

Abaixo estão os scripts utilizados para criar o banco de dados no **schema luccagomes** do Oracle.

*-- -------------------------------------------------------------------------*

*-- Criação das tabelas no schema luccagomes*

*-- -------------------------------------------------------------------------*

*--------------------------------------------------------------------------*

*-- TABELA USUARIO*

*--------------------------------------------------------------------------*

CREATE TABLE luccagomes.usuario (

id *NUMBER* GENERATED ALWAYS AS IDENTITY,

nome *VARCHAR2*(255) NOT NULL,

login *VARCHAR2*(255) NOT NULL,

senha *VARCHAR2*(255) NOT NULL,

nasc *DATE* NOT NULL,

ativo *CHAR*(1) DEFAULT 'Y' NOT NULL,

*-- 'Y' = ativo, 'N' = inativo*

CONSTRAINT pk\_usuario PRIMARY KEY (id),

CONSTRAINT uk\_usuario\_login UNIQUE (login)

);

*--------------------------------------------------------------------------*

*-- TABELA IDIOMA*

*--------------------------------------------------------------------------*

CREATE TABLE luccagomes.idioma (

id *NUMBER* GENERATED ALWAYS AS IDENTITY,

idioma *VARCHAR2*(24) NOT NULL,

CONSTRAINT pk\_idioma PRIMARY KEY (id),

CONSTRAINT uk\_idioma UNIQUE (idioma)

);

*--------------------------------------------------------------------------*

*-- TABELA GENERO*

*--------------------------------------------------------------------------*

CREATE TABLE luccagomes.genero (

id *NUMBER* GENERATED ALWAYS AS IDENTITY,

genero *VARCHAR2*(24) NOT NULL,

CONSTRAINT pk\_genero PRIMARY KEY (id),

CONSTRAINT uk\_genero UNIQUE (genero)

);

*--------------------------------------------------------------------------*

*-- TABELA MIDIA*

*--------------------------------------------------------------------------*

CREATE TABLE luccagomes.midia (

id *NUMBER* GENERATED ALWAYS AS IDENTITY,

titulo *VARCHAR2*(255) NOT NULL,

sinopse *CLOB*,

avaliacao *NUMBER*(3,2),

poster *VARCHAR2*(255),

atores *VARCHAR2*(255),

dt\_lancamento *DATE* NOT NULL,

valor *NUMBER*(10,2) NOT NULL,

duracao *NUMBER*, *-- exclusivo de filme*

temporadas *NUMBER*, *-- exclusivo de série*

ativo *CHAR*(1) DEFAULT 'Y' NOT NULL,

*-- 'Y' = ativo, 'N' = inativo*

CONSTRAINT pk\_midia PRIMARY KEY (id),

CONSTRAINT ck\_filme\_ou\_serie CHECK (

(

duracao IS NULL

AND temporadas IS NOT NULL

)

OR

(

duracao IS NOT NULL

AND temporadas IS NULL

)

)

);

*--------------------------------------------------------------------------*

*-- TABELA IDIOMAS\_DA\_MIDIA*

*--------------------------------------------------------------------------*

CREATE TABLE luccagomes.idiomas\_da\_midia (

midia\_id *NUMBER*,

idioma\_id *NUMBER*,

CONSTRAINT pk\_idiomas\_da\_midia PRIMARY KEY (midia\_id, idioma\_id),

CONSTRAINT fk\_midia\_tem\_idioma FOREIGN KEY (midia\_id)

REFERENCES luccagomes.midia(id),

CONSTRAINT fk\_idioma\_da\_midia FOREIGN KEY (idioma\_id)

REFERENCES luccagomes.idioma(id)

);

*--------------------------------------------------------------------------*

*-- TABELA GENEROS\_DA\_MIDIA*

*--------------------------------------------------------------------------*

CREATE TABLE luccagomes.generos\_da\_midia (

midia\_id *NUMBER*,

genero\_id *NUMBER*,

CONSTRAINT pk\_generos\_da\_midia PRIMARY KEY (midia\_id, genero\_id),

CONSTRAINT fk\_midia\_tem\_genero FOREIGN KEY (midia\_id)

REFERENCES luccagomes.midia(id),

CONSTRAINT fk\_genero\_da\_midia FOREIGN KEY (genero\_id)

REFERENCES luccagomes.genero(id)

);

*--------------------------------------------------------------------------*

*-- TABELA NOTA\_FISCAL*

*--------------------------------------------------------------------------*

CREATE TABLE luccagomes.nota\_fiscal (

usuario\_id *NUMBER*,

valor\_total *NUMBER*(10,2) NOT NULL,

dt\_pagamento *TIMESTAMP* DEFAULT *SYSTIMESTAMP*,

CONSTRAINT pk\_nota\_fiscal PRIMARY KEY (usuario\_id, dt\_pagamento),

CONSTRAINT fk\_usuario\_possui FOREIGN KEY (usuario\_id)

REFERENCES luccagomes.usuario(id)

);

*--------------------------------------------------------------------------*

*-- TABELA ALUGUEL*

*--------------------------------------------------------------------------*

CREATE TABLE luccagomes.aluguel (

usuario\_id *NUMBER*,

midia\_id *NUMBER*,

dt\_inicio *TIMESTAMP* DEFAULT *SYSTIMESTAMP*,

dt\_expira *TIMESTAMP* DEFAULT (*SYSTIMESTAMP* + INTERVAL '30' DAY) NOT NULL,

valor *NUMBER*(10,2) NOT NULL,

CONSTRAINT pk\_aluguel PRIMARY KEY (usuario\_id, midia\_id, dt\_inicio),

CONSTRAINT fk\_usuario\_alugou FOREIGN KEY (usuario\_id, dt\_inicio)

REFERENCES luccagomes.nota\_fiscal (usuario\_id, dt\_pagamento),

CONSTRAINT fk\_midia\_foi\_alugada FOREIGN KEY (midia\_id)

REFERENCES luccagomes.midia (id),

CONSTRAINT ck\_dt\_aluguel CHECK (dt\_inicio < dt\_expira),

CONSTRAINT ck\_valor\_aluguel\_positivo CHECK (valor > 0)

);

*--------------------------------------------------------------------------*

*-- TABELA COMPRA*

*--------------------------------------------------------------------------*

CREATE TABLE luccagomes.compra (

usuario\_id *NUMBER*,

midia\_id *NUMBER*,

dt\_compra *TIMESTAMP*,

valor *NUMBER*(10,2) NOT NULL,

CONSTRAINT pk\_comprou PRIMARY KEY (usuario\_id, midia\_id),

CONSTRAINT fk\_usuario\_comprou FOREIGN KEY (usuario\_id, dt\_compra)

REFERENCES luccagomes.nota\_fiscal (usuario\_id, dt\_pagamento),

CONSTRAINT fk\_midia\_foi\_comprada FOREIGN KEY (midia\_id)

REFERENCES luccagomes.midia (id),

CONSTRAINT ck\_valor\_compra\_positivo CHECK (valor > 0)

);

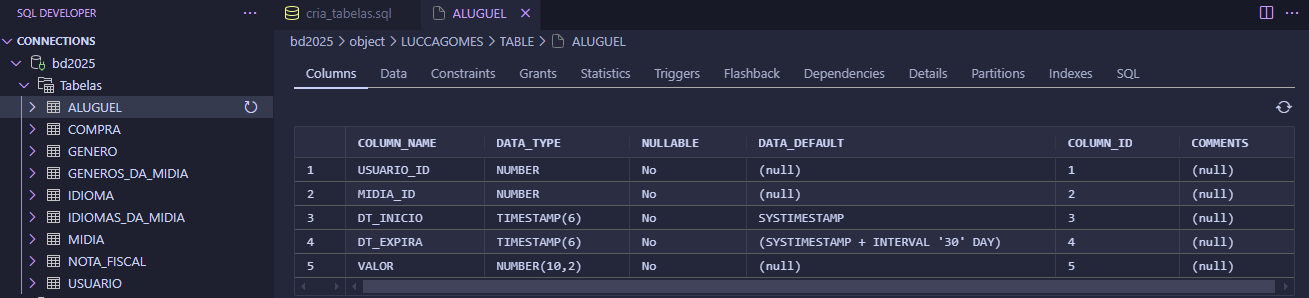
### **1.2. Relatório**

O script em acima representa a adaptação para o Oracle do banco de dados originalmente desenvolvido em PostgreSQL, como parte do trabalho realizado na disciplina de Banco de Dados I no semestre passado ( [Link do sql do projeto de BD 1](https://github.com/LuccaGiovane/UEL-BD/blob/main/marketplace.sql) ). Durante o processo de migração, algumas mudanças foram feitas para atender às especificidades do Oracle, como ajustes na definição de chaves primárias e estrangeiras, além de modificações nas restrições de integridade dos dados.

Apesar dessas mudanças, o script preserva integralmente a lógica e os objetivos estabelecidos no projeto original, garantindo que a estrutura e as funcionalidades previamente definidas continuem a operar conforme o planejado.

### **1.3 Tabelas após executar o script de criação**

Abaixo um exemplo mostrando a tabela *ALUGUEL* criada após rodar o script. As demais tabelas foram criadas como o esperado e seus nomes estão ao lado esquerdo no print.



## **2. Procedimento em PL/SQL para Carga de Dados (Fator de Escala)**

### **2.1. Script de Criação do Procedimento**

*-- carrega\_dados.sql*

*-------------------------------------------------------------------------------*

*-- Script que:*

*-- 1) Cria/recria o procedimento sp\_carga\_dados*

*-- 2) Executa sp\_carga\_dados(p\_fator)*

*-------------------------------------------------------------------------------*

*PROMPT* ========== CRIANDO/RECRIANDO O PROCEDIMENTO sp\_carga\_dados ==========

CREATE OR REPLACE PROCEDURE luccagomes.sp\_carga\_dados(p\_fator *IN* *NUMBER*) AS

BEGIN

*------------------------------------------------------------------------------*

*-- Bloco de limpeza: Deleta os registros das tabelas na ordem correta*

*------------------------------------------------------------------------------*

*-- Apaga os dados das tabelas-filho primeiro para não violar as restrições:*

*EXECUTE* IMMEDIATE 'DELETE FROM luccagomes.compra';

*EXECUTE* IMMEDIATE 'DELETE FROM luccagomes.aluguel';

*EXECUTE* IMMEDIATE 'DELETE FROM luccagomes.nota\_fiscal';

*EXECUTE* IMMEDIATE 'DELETE FROM luccagomes.idiomas\_da\_midia';

*EXECUTE* IMMEDIATE 'DELETE FROM luccagomes.generos\_da\_midia';

*EXECUTE* IMMEDIATE 'DELETE FROM luccagomes.midia';

*EXECUTE* IMMEDIATE 'DELETE FROM luccagomes.usuario';

*EXECUTE* IMMEDIATE 'DELETE FROM luccagomes.idioma';

*EXECUTE* IMMEDIATE 'DELETE FROM luccagomes.genero';

COMMIT;

*------------------------------------------------------------------------------*

*-- Bloco de inserção de dados*

*------------------------------------------------------------------------------*

DECLARE

v\_count *NUMBER* := 1;

v\_limite *NUMBER* := 5 \* p\_fator; *-- base de 5, ajustada pelo fator, para mudar o valor do fator é la em baixo nesse script*

*-- Variáveis auxiliares para compra*

v\_midia\_compra *NUMBER*;

v\_count\_compra *NUMBER*;

*-- Variáveis auxiliares para idiomas*

v\_idioma1 luccagomes.idioma.id%TYPE;

v\_idioma2 luccagomes.idioma.id%TYPE;

BEGIN

*----------------------------------------------------------------------*

*-- 2) Inserir usuários*

*----------------------------------------------------------------------*

WHILE v\_count <= v\_limite LOOP

INSERT INTO luccagomes.usuario (nome, login, senha, nasc, ativo)

VALUES (

'Usuario ' || v\_count,

'login' || v\_count,

'senha' || v\_count,

*TRUNC*(*TO\_DATE*('01/01/1990','DD/MM/YYYY') + DBMS\_RANDOM.value(1,10000)), *-- datas aleatórias*

CASE WHEN *MOD*(v\_count, 2) = 0 THEN 'Y' ELSE 'N' END

);

v\_count := v\_count + 1;

END LOOP;

*----------------------------------------------------------------------*

*-- 3) Inserir idiomas (20 registros)*

*----------------------------------------------------------------------*

FOR *i* *IN* 1..20 LOOP

INSERT INTO luccagomes.idioma (idioma)

VALUES ('Idioma\_' || *i*);

END LOOP;

*----------------------------------------------------------------------*

*-- 4) Inserir gêneros (20 registros)*

*----------------------------------------------------------------------*

FOR *i* *IN* 1..20 LOOP

INSERT INTO luccagomes.genero (genero)

VALUES ('Genero\_' || *i*);

END LOOP;

*----------------------------------------------------------------------*

*-- 5) Inserir mídias*

*----------------------------------------------------------------------*

v\_count := 1;

WHILE v\_count <= v\_limite LOOP

INSERT INTO luccagomes.midia

(titulo, sinopse, avaliacao, poster, atores, dt\_lancamento,

valor, duracao, temporadas, ativo)

VALUES (

'Midia ' || v\_count,

'Sinopse da midia ' || v\_count,

*ROUND*(DBMS\_RANDOM.value(0,10),2),

'poster' || v\_count || '.jpg',

'Ator\_' || v\_count || ', Atriz\_' || v\_count,

*TRUNC*(*TO\_DATE*('01/01/2000','DD/MM/YYYY') + DBMS\_RANDOM.value(1,5000)), *-- datas aleatórias*

*ROUND*(DBMS\_RANDOM.value(10,100),2),

CASE WHEN *MOD*(v\_count,2)=0 THEN *ROUND*(DBMS\_RANDOM.value(90,180))

ELSE NULL END, *-- Se par, é filme (duracao != NULL)*

CASE WHEN *MOD*(v\_count,2)=1 THEN *TRUNC*(DBMS\_RANDOM.value(1,6))

ELSE NULL END, *-- Se ímpar, é série (temporadas != NULL)*

CASE WHEN *MOD*(v\_count,3)=0 THEN 'N' ELSE 'Y' END

);

v\_count := v\_count + 1;

END LOOP;

*----------------------------------------------------------------------*

*-- 6) Inserir relacionamentos entre mídias e idiomas*

*----------------------------------------------------------------------*

FOR mid *IN* (SELECT id FROM luccagomes.midia) LOOP

*-- Seleciona e insere o primeiro idioma aleatório para cada mídia*

SELECT id

INTO v\_idioma1

FROM (SELECT id FROM luccagomes.idioma ORDER BY DBMS\_RANDOM.value)

WHERE ROWNUM = 1;

INSERT INTO luccagomes.idiomas\_da\_midia (midia\_id, idioma\_id)

VALUES (mid.id, v\_idioma1);

*-- Chance aleatória de inserir um segundo idioma*

IF DBMS\_RANDOM.value(0,1) > 0.5 THEN

*-- Seleciona um segundo idioma que não seja o primeiro*

SELECT id

INTO v\_idioma2

FROM (SELECT id FROM luccagomes.idioma WHERE id <> v\_idioma1 ORDER BY DBMS\_RANDOM.value)

WHERE ROWNUM = 1;

INSERT INTO luccagomes.idiomas\_da\_midia (midia\_id, idioma\_id)

VALUES (mid.id, v\_idioma2);

END IF;

END LOOP;

*----------------------------------------------------------------------*

*-- 7) Inserir relacionamentos entre mídias e gêneros*

*----------------------------------------------------------------------*

FOR mid *IN* (SELECT id FROM luccagomes.midia) LOOP

DECLARE

v\_genero1 luccagomes.genero.id%TYPE;

v\_genero2 luccagomes.genero.id%TYPE;

BEGIN

*-- Seleciona e insere o primeiro gênero aleatório para cada mídia*

SELECT id

INTO v\_genero1

FROM (SELECT id FROM luccagomes.genero ORDER BY DBMS\_RANDOM.value)

WHERE ROWNUM = 1;

INSERT INTO luccagomes.generos\_da\_midia (midia\_id, genero\_id)

VALUES (mid.id, v\_genero1);

*-- Chance aleatória de inserir um segundo gênero*

IF DBMS\_RANDOM.value(0,1) > 0.5 THEN

*-- Seleciona um segundo gênero que não seja o primeiro*

SELECT id

INTO v\_genero2

FROM (SELECT id FROM luccagomes.genero WHERE id <> v\_genero1 ORDER BY DBMS\_RANDOM.value)

WHERE ROWNUM = 1;

INSERT INTO luccagomes.generos\_da\_midia (midia\_id, genero\_id)

VALUES (mid.id, v\_genero2);

END IF;

END;

END LOOP;

*----------------------------------------------------------------------*

*-- 8) Inserir notas fiscais (2 por usuário ativo)*

*----------------------------------------------------------------------*

FOR usr *IN* (SELECT id FROM luccagomes.usuario WHERE ativo = 'Y') LOOP

INSERT INTO luccagomes.nota\_fiscal (usuario\_id, valor\_total)

VALUES (usr.id, *ROUND*(DBMS\_RANDOM.value(50,300),2));

INSERT INTO luccagomes.nota\_fiscal (usuario\_id, valor\_total)

VALUES (usr.id, *ROUND*(DBMS\_RANDOM.value(50,300),2));

END LOOP;

*----------------------------------------------------------------------*

*-- 9) Inserir alugueis e compras para cada nota fiscal*

*----------------------------------------------------------------------*

FOR nf *IN* (

SELECT usuario\_id, dt\_pagamento, valor\_total

FROM luccagomes.nota\_fiscal

) LOOP

*-- 9.1) Aluguel (chance ~50%)*

IF DBMS\_RANDOM.value(0,1) > 0.5 THEN

INSERT INTO luccagomes.aluguel

(usuario\_id, midia\_id, dt\_inicio, dt\_expira, valor)

VALUES (

nf.usuario\_id,

(SELECT id FROM (SELECT id FROM luccagomes.midia ORDER BY DBMS\_RANDOM.value) WHERE ROWNUM = 1),

nf.dt\_pagamento,

nf.dt\_pagamento + *NUMTODSINTERVAL*(3,'DAY'),

*ROUND*(DBMS\_RANDOM.value(10,50),2)

);

END IF;

*-- 9.2) Compra (chance ~70%)*

IF DBMS\_RANDOM.value(0,1) > 0.3 THEN

*-- Escolhe aleatoriamente uma mídia*

SELECT id

INTO v\_midia\_compra

FROM (

SELECT id

FROM luccagomes.midia

ORDER BY DBMS\_RANDOM.value

)

WHERE ROWNUM = 1;

*-- Verifica se esse usuário já comprou essa mídia*

SELECT *COUNT*(\*)

INTO v\_count\_compra

FROM luccagomes.compra

WHERE usuario\_id = nf.usuario\_id

AND midia\_id = v\_midia\_compra;

*-- Se ainda não comprou, faz a compra*

IF v\_count\_compra = 0 THEN

INSERT INTO luccagomes.compra

(usuario\_id, midia\_id, dt\_compra, valor)

VALUES (

nf.usuario\_id,

v\_midia\_compra,

nf.dt\_pagamento,

*ROUND*(DBMS\_RANDOM.value(15,80),2)

);

END IF;

END IF;

END LOOP;

COMMIT;

END;

END;

/

*SHOW* ERRORS;

*-------------------------------------------------------------------------------*

*-- Invoca a Procedure*

*-------------------------------------------------------------------------------*

*PROMPT* ========== EXECUTANDO sp\_carga\_dados(p\_fator) ==========

BEGIN

*-- Ajuste aqui o fator da escala: 1, 5 etc.*

luccagomes.sp\_carga\_dados(5); *-- Exemplo: p\_fator = 5*

END;

/

### **2.2. Relatório**

O script acima começa fazendo algumas deleções dos valores pois como rodei varias vezes testando ajudou muito e até para testar outras coisas.

Iniciando as inserções nas tabelas *USUARIOS*, *IDIOMAS* e *GENEROS* são gerados valores genéricos como: ‘usuário N’, ‘idioma\_N’, ‘genero\_N’ sendo ‘N’ um valor numérico (1, 2, 3, ... , N). Além disso, existe uma chance de a mídia ter mais de um gênero e mais de um idioma.

Agora sobre transações (*NOTA\_FISCAL*, *ALUGUEL* e *COMPRA*) esta sendo populado da seguinte forma:

* Notas Fiscais: Para cada usuário ativo, gera duas notas fiscais com valores aleatórios
* Aluguéis e Compras: Para cada nota fiscal, tem uma chance de registrar um aluguel (com data de início e expiração) e uma compra só se o usuário ainda não tiver comprado aquela mídia pois não faz sentido ele querer alugar algo que ele ja comprou.

### **2.3 Exemplos de tabelas após executar o script de popular**

Tabela *USUARIO*



As tabelas *MIDIA* e *IDIOMA* se relacionam através da tabela *IDIOMAS\_DA\_MIDIA* (o mesmo acontece com os gêneros com *GENERO* e *GENERO\_DA\_MIDIA*) abaixo o exemplo dessas três tabelas

Tabela *MIDIA*

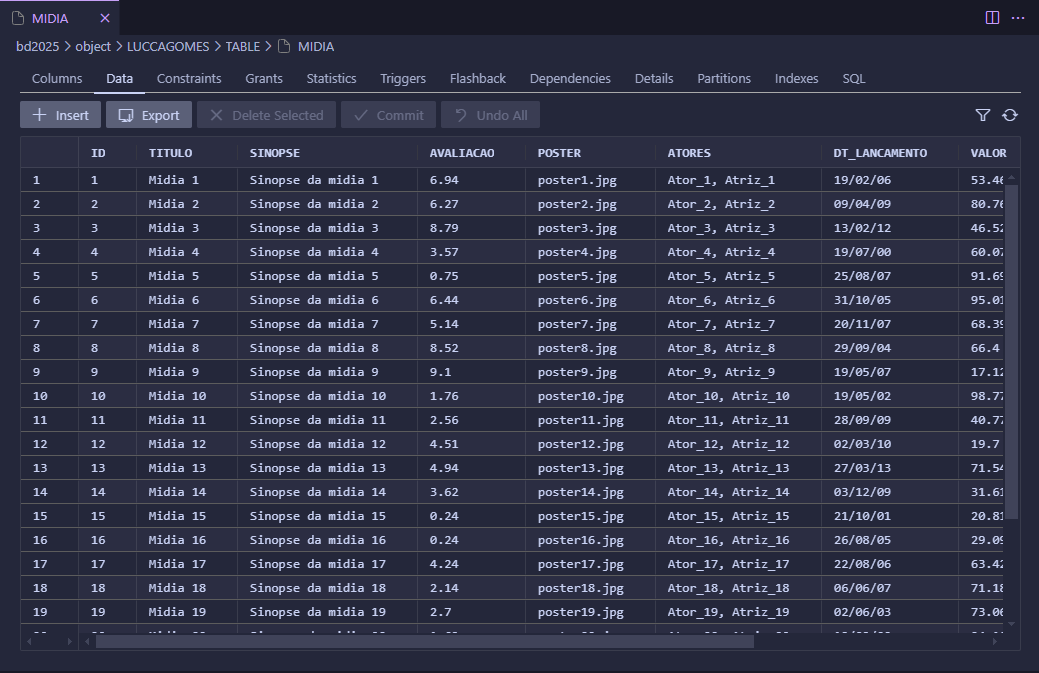


Tabela *IDIOMA*

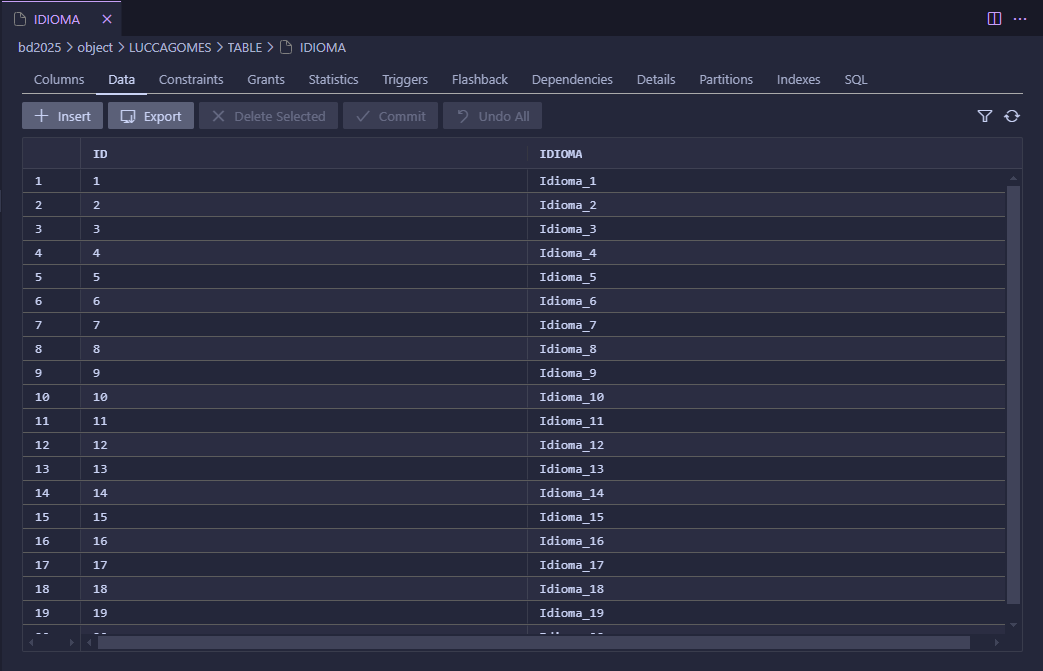
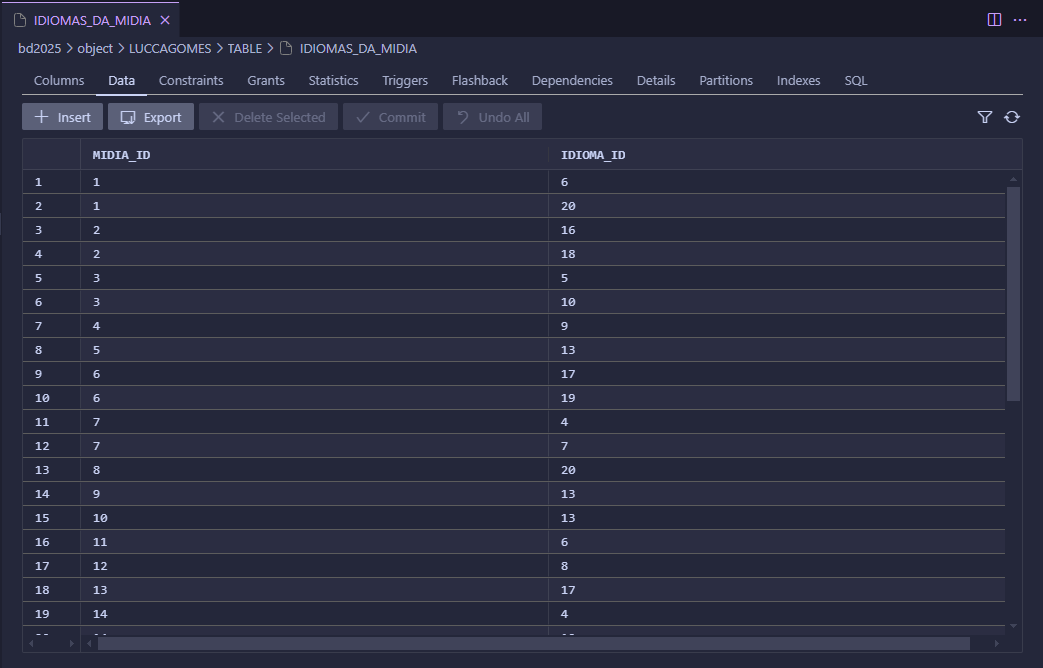


Tabela *IDIOMAS\_DA\_MIDIA*



## **3. Implementação de uma visão computada e uma visão materializada**

### **3.1. Script de Criação das Visões**

*PROMPT* ========== CRIANDO VIEW (VISÃO COMPUTADA) ==========

CREATE OR REPLACE VIEW luccagomes.vw\_midia\_ativa AS

SELECT

id,

titulo,

dt\_lancamento,

valor,

avaliacao,

poster,

atores

FROM

luccagomes.midia

WHERE

ativo = 'Y';

*PROMPT* ========== EXEMPLO DE USO DA VIEW (CONSULTA) ==========

*-- Verificando quantas mídias ativas temos no momento*

SELECT *COUNT*(\*) AS total\_midia\_ativa

FROM luccagomes.vw\_midia\_ativa;

*PROMPT* ========== FAZENDO ATUALIZACAO QUE AFETA *A* VIEW ==========

DECLARE

v\_id\_midia *NUMBER*;

BEGIN

*-- Pegamos 1 mídia que esteja ativa*

SELECT id

INTO v\_id\_midia

FROM luccagomes.midia

WHERE ativo = 'Y'

AND ROWNUM = 1; *-- pega a primeira que encontrar*

*-- Desativa essa mídia (muda de Y para N)*

UPDATE luccagomes.midia

*SET* ativo = 'N'

WHERE id = v\_id\_midia;

COMMIT;

DBMS\_OUTPUT.put\_line(

'Midia ' || v\_id\_midia || ' foi desativada. Verifique a vw\_midia\_ativa.'

);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.put\_line('Nenhuma mídia ativa encontrada para desativar.');

END;

/

*-- Agora se consultar novamente a view, essa mídia não aparecerá mais*

SELECT id, titulo

FROM luccagomes.vw\_midia\_ativa

WHERE ROWNUM <= 5; *-- Exibe até as 5 primeiras mídias ativas*

*PROMPT* ========== CRIANDO/RECRIANDO MATERIALIZED VIEW (SEM FILTRO) ==========

BEGIN

*EXECUTE* IMMEDIATE 'DROP MATERIALIZED VIEW luccagomes.mv\_compras\_resumo';

EXCEPTION

WHEN OTHERS THEN

IF SQLCODE != -12003 THEN *-- ORA-12003: materialized view does not exist*

RAISE;

END IF;

END;

/

CREATE MATERIALIZED VIEW luccagomes.mv\_compras\_resumo

BUILD IMMEDIATE

REFRESH ON DEMAND

AS

SELECT

*c*.usuario\_id,

*COUNT*(\*) AS total\_compras,

*SUM*(*c*.valor) AS valor\_total

FROM

luccagomes.compra *c*

GROUP BY

*c*.usuario\_id;

*PROMPT* ========== EXEMPLO DE USO DA MATERIALIZED VIEW ==========

*-- Consulta a MV para ver quantas compras por usuário*

SELECT \* FROM luccagomes.mv\_compras\_resumo

ORDER BY usuario\_id;

*PROMPT* ========== EXEMPLO DE ATUALIZACAO QUE AFETA O CONTEUDO DA MV ==========

DECLARE

v\_usuario *NUMBER*;

v\_dt\_pag *TIMESTAMP*;

v\_midia *NUMBER*;

BEGIN

*-- Pegamos algum usuario\_id e dt\_pagamento que existam em nota\_fiscal*

SELECT usuario\_id, dt\_pagamento

INTO v\_usuario, v\_dt\_pag

FROM luccagomes.nota\_fiscal

WHERE ROWNUM = 1;

*-- Pegamos alguma midia que ESTE USUÁRIO ainda NÃO comprou*

SELECT m.id

INTO v\_midia

FROM luccagomes.midia m

WHERE ROWNUM = 1

AND NOT EXISTS (

SELECT 1

FROM luccagomes.compra *c*

WHERE *c*.usuario\_id = v\_usuario

AND *c*.midia\_id = m.id

);

*-- Insere uma nova compra na tabela base (compra),*

*-- usando esse usuário, essa mídia e a data de pagamento acima*

INSERT INTO luccagomes.compra (usuario\_id, midia\_id, dt\_compra, valor)

VALUES (v\_usuario, v\_midia, v\_dt\_pag, 40);

COMMIT;

DBMS\_OUTPUT.put\_line('Inserida compra do usuario ' || v\_usuario

|| ' na midia ' || v\_midia

|| ' no dt\_pagamento ' || v\_dt\_pag

|| ' com valor R$40');

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.put\_line(

'Não foi possível inserir compra: não há nota\_fiscal, midia ou ' ||

'todas as mídias já foram compradas por esse usuário.'

);

END;

/

*-- Consulta novamente a MV (sem REFRESH) para ver se já atualizou*

*-- (não vai atualizar até fazermos o REFRESH, pois é "REFRESH ON DEMAND")*

SELECT \* FROM luccagomes.mv\_compras\_resumo

ORDER BY usuario\_id;

*PROMPT* ========== REALIZANDO O REFRESH MANUAL DA MV ==========

*-- forçar o refresh para a MV recalcular.*

*EXEC* DBMS\_MVIEW.REFRESH('luccagomes.mv\_compras\_resumo');

*-- Consulta após o REFRESH*

SELECT \* FROM luccagomes.mv\_compras\_resumo

ORDER BY usuario\_id;

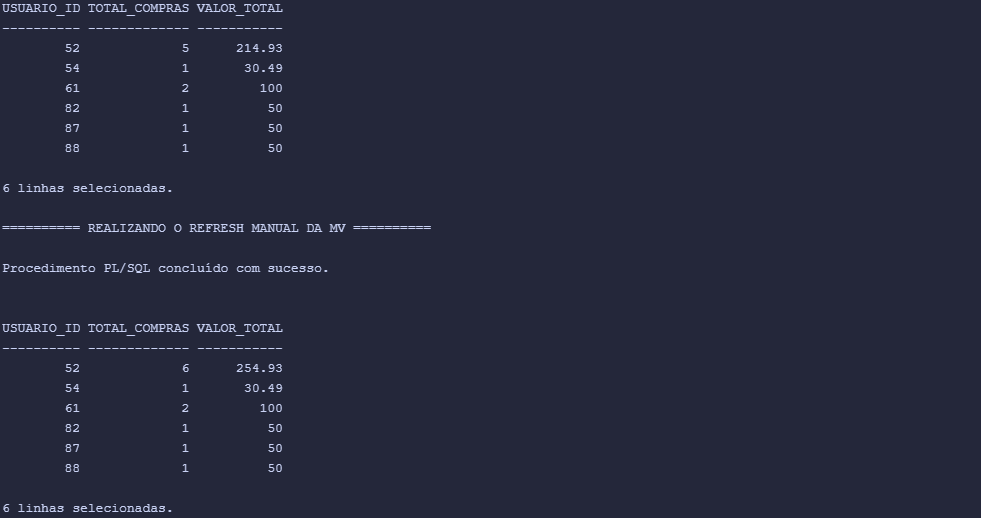
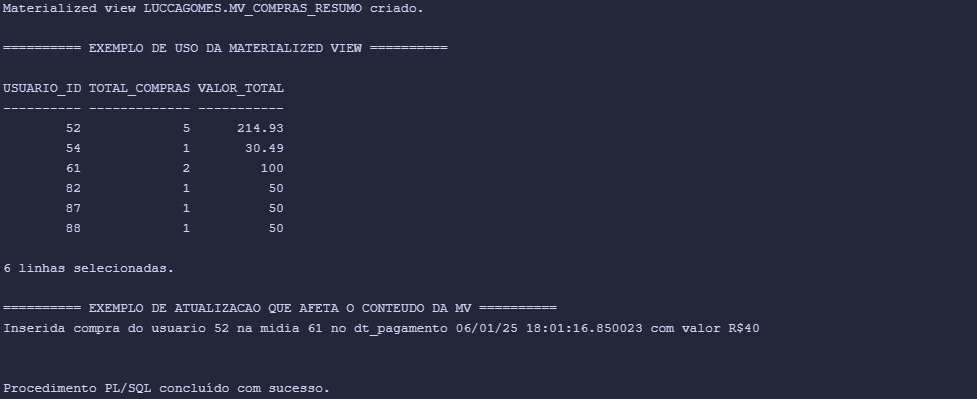
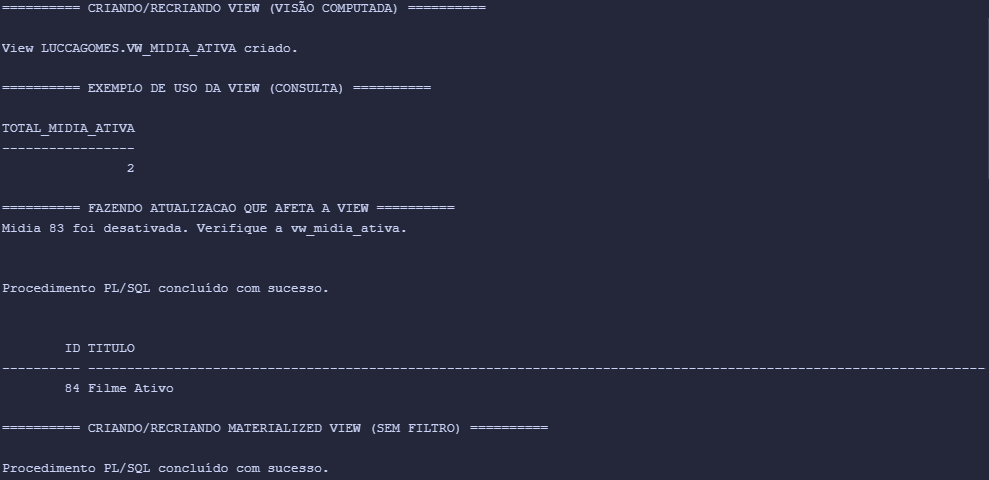
/

### **3.2. Relatório**

View computada (vw\_midia\_ativa): a ideia e montar uma view que mostra as mídias ativas (ativo = Y na table *MIDIA*). Uma consulta vai contar quantas mídias estão ativas e um bloco PL/SQL vai selecionar uma mídia ativa, desativar ela (vai mudar o ativo para N na table) e depois a view deixa de exibir essa mídia.

View materializada (mv\_compras\_resumo): vai criar uma materialized view que resume as compras por usuário: total e a soma dos valores. Uma consulta mostra o resumo das compras e um bloco PL/SQL vai inserir uma nova compra caso o usuário ainda não tiver comprado essa mídia, porém essa mídia não aparece nessa view ate que seja feito um refresh manual 'DBMS\_MVIEW.REFRESH'

### **3.3 Output**



## **4. Implementação de Common Table Expressions**

### **4.1. Script de Criação das CTE**

*PROMPT* ========== CONSULTA NAO RECURSIVA (WITH) ==========

WITH

cte\_gen\_count AS (

SELECT

g.genero,

*COUNT*(gdm.midia\_id) AS total\_midia

FROM

luccagomes.genero g

LEFT JOIN luccagomes.generos\_da\_midia gdm

ON gdm.genero\_id = g.id

GROUP BY

g.genero

),

cte\_idioma\_count AS (

SELECT

*i*.idioma,

*COUNT*(idm.midia\_id) AS total\_midia

FROM

luccagomes.idioma *i*

LEFT JOIN luccagomes.idiomas\_da\_midia idm

ON idm.idioma\_id = *i*.id

GROUP BY

*i*.idioma

)

SELECT

'Gênero' AS tipo,

genero AS nome,

total\_midia

FROM

(

SELECT genero, total\_midia

FROM cte\_gen\_count

ORDER BY total\_midia *DESC*

FETCH *FIRST* 3 ROWS ONLY *-- top 3 gêneros*

)

UNION ALL

SELECT

'Idioma' AS tipo,

idioma AS nome,

total\_midia

FROM

(

SELECT idioma, total\_midia

FROM cte\_idioma\_count

ORDER BY total\_midia *DESC*

FETCH *FIRST* 3 ROWS ONLY *-- top 3 idiomas*

)

ORDER BY

tipo *DESC*,

total\_midia *DESC*;

*PROMPT* ========== FIM DA CONSULTA NAO RECURSIVA ==========

*PROMPT*

*PROMPT* ========== CONSULTA RECURSIVA (WITH RECURSIVE) ==========

WITH RECURSIVE cte\_calendario (dia, dia\_fim) AS (

*-- ancora: pega a data mínima de dt\_inicio e a data máxima de dt\_inicio*

SELECT

*TRUNC*(*MIN*(*a*.dt\_inicio)) AS dia,

*TRUNC*(*MAX*(*a*.dt\_inicio)) AS dia\_fim

FROM

luccagomes.aluguel *a*

UNION ALL

*-- passo recursivo: avança o dia + 1, até alcançar dia\_fim*

SELECT

dia + 1,

dia\_fim

FROM

cte\_calendario

WHERE

(dia + 1) <= dia\_fim

)

SELECT

*c*.dia,

*COUNT*(*a*.usuario\_id) AS total\_alugueis\_no\_dia

FROM

cte\_calendario *c*

LEFT JOIN luccagomes.aluguel *a*

ON *TRUNC*(*a*.dt\_inicio) = *c*.dia

GROUP BY

*c*.dia

ORDER BY

*c*.dia;

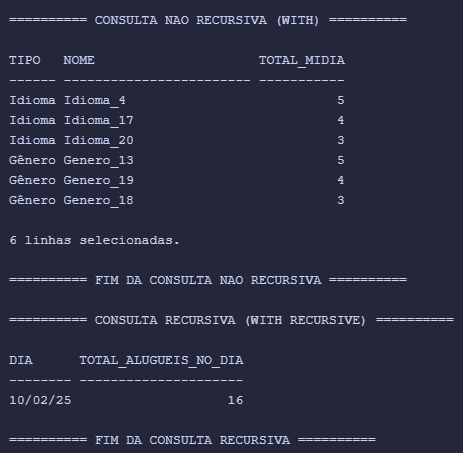
*PROMPT* ========== FIM DA CONSULTA RECURSIVA ==========

### **4.2. Relatório**

Consulta não recursiva: esta definindo duas CTE's para contar as mídias por gênero e idioma. Depois seleciona s três gêneros e idiomas com mais mídias e combina os resultados.

Consulta recursiva: gera um calendário de datas começando do dia mínimo ate o máximo que esta registrado em dt\_inicio nos aluguéis e, para cada data vai ser contado quantos aluguéis começaram naquele dia e vai agrupar e ordenar os resultados.

### **4.3. Output**



## **5. Implementação de Window Functions**

### **5.1. Script de Criação das Window Functions**

*PROMPT* ========== CONSULTA 1: RANKING DE MÍDIAS MAIS COMPRADAS ==========

SELECT

midia\_id,

total\_por\_midia,

*RANK*() OVER (ORDER BY total\_por\_midia *DESC*) AS rank\_por\_valor

FROM (

SELECT

*c*.midia\_id,

*SUM*(*c*.valor) AS total\_por\_midia

FROM

luccagomes.compra *c*

GROUP BY

*c*.midia\_id

)

ORDER BY

rank\_por\_valor;

*PROMPT* ========== CONSULTA 2: DIFERENÇA ENTRE COMPRAS CONSECUTIVAS ==========

SELECT

usuario\_id,

*TO\_CHAR*(dt\_compra, 'DD/MM/YYYY HH24:MI:SS') AS dt\_compra,

*SUM*(valor) AS total\_compra,

*RANK*() OVER (ORDER BY *SUM*(valor) *DESC*) AS rank\_global

FROM

luccagomes.compra

GROUP BY

usuario\_id,

dt\_compra

ORDER BY

rank\_global;

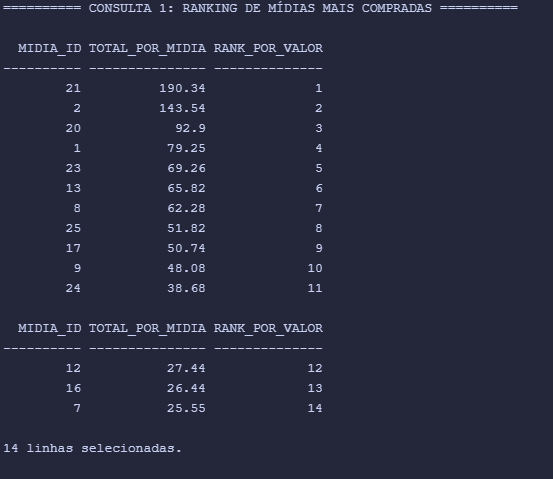
*PROMPT* ========== FIM DO SCRIPT DE WINDOW FUNCTIONS ==========

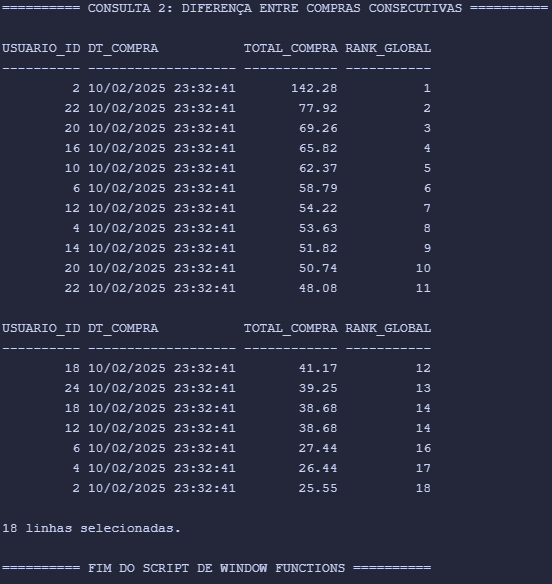
### **5.2 Relatório**

A ideia da primeira consulta é classificar as mídias com base no valor total acumulado de compras. Ela faz uma subconsulta interna que calcula o total das vendas para cada mídia, soma o valor de todas as compras daquela mídia mantendo todas as linhas da tabla compra. Depois ela aplica um RANK() que classifica as mídias em ordem decrescente para gerar o ranking.

A segunda consulta analisa o comportamento de compra dos usuários, mostrando quanto tempo passa entre uma compra e outra. Ela compara a data de cada compra com a data da compra anterior do mesmo usuário. A diferença entre as duas datas é calculada em segundos.

### **5.3 Output**





## **6. Implementação de uma Função SQL**

### **6.1 Script de Criação da Função SQL**

*PROMPT* ========== CRIAÇÃO DA FUNÇÃO: FN\_TOTAL\_COMPRAS ==========

CREATE OR REPLACE FUNCTION luccagomes.fn\_total\_compras (

p\_usuario\_id *IN* *NUMBER*

)

RETURN *NUMBER*

IS

v\_total *NUMBER*(10,2);

BEGIN

SELECT *NVL*(*SUM*(*c*.valor), 0)

INTO v\_total

FROM luccagomes.compra *c*

WHERE *c*.usuario\_id = p\_usuario\_id;

RETURN v\_total;

END;

/

*PROMPT* ========== CONSULTA QUE INVOCA *A* FUNÇÃO FN\_TOTAL\_COMPRAS ==========

SELECT

u.id AS usuario\_id,

*SUBSTR*(u.nome, 1, 20) AS nome\_usuario,

luccagomes.fn\_total\_compras(u.id) AS total\_gasto

FROM

luccagomes.usuario u

ORDER BY

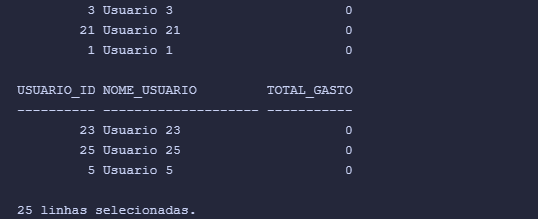
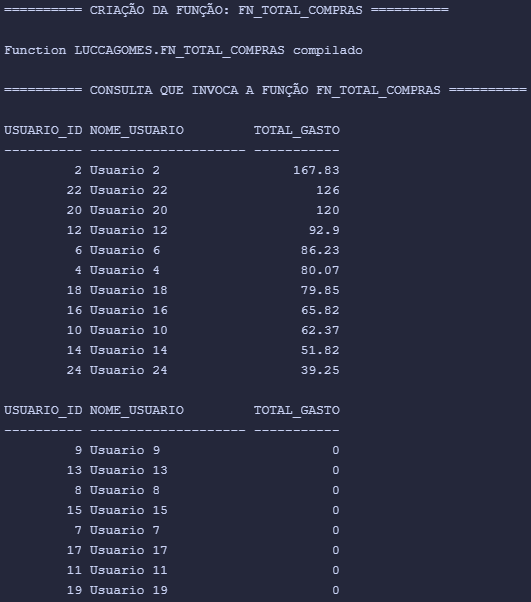
total\_gasto *DESC*;

### **6.2 Relatório**

A função fn\_total\_compras calcula quanto dinheiro cada usuário gastou em compras. Ela pega o ID de um usuário e verifica todas as compras que ele fez, somando os valores. Se o usuário não fez nenhuma compra, ela retorna 0.

A consulta usa essa função para mostrar uma lista de todos os usuários, exibindo o ID, o nome (limitado a 20 caracteres para facilitar a leitura) e o total gasto por cada um. A lista é ordenada do maior gastador para o menor.

### **6.3 Output**



## **7. Implementação de um Trigger DML**

### **7.1 Script de Criação do Trigger DML**

*PROMPT* ========== INÍCIO DOS TRIGGERS E TESTES (DML.sql) ==========

*-------------------------------------------------------------------------------*

*PROMPT* 1) CRIAÇÃO DO TRIGGER PARA AJUSTE AUTOMÁTICO DO VALOR DE COMPRA

*-------------------------------------------------------------------------------*

CREATE OR REPLACE TRIGGER luccagomes.trg\_auto\_valor\_compra

BEFORE INSERT ON luccagomes.compra

FOR EACH ROW

DECLARE

v\_valor *NUMBER*(10,2);

BEGIN

*-- Se o valor da compra não for informado (ou for nulo), usar o valor da mídia*

IF :NEW.valor IS NULL THEN

SELECT m.valor

INTO v\_valor

FROM luccagomes.midia m

WHERE m.id = :NEW.midia\_id;

:NEW.valor := v\_valor;

END IF;

END;

/

*PROMPT* ... Trigger luccagomes.trg\_auto\_valor\_compra criado com sucesso.

*-------------------------------------------------------------------------------*

*PROMPT* 2) CRIAÇÃO DO TRIGGER QUE IMPEDE COMPRA DE MÍDIA INATIVA

*-------------------------------------------------------------------------------*

CREATE OR REPLACE TRIGGER luccagomes.trg\_check\_midia\_ativa\_compra

BEFORE INSERT OR UPDATE ON luccagomes.compra

FOR EACH ROW

DECLARE

v\_ativo *CHAR*(1);

BEGIN

SELECT m.ativo

INTO v\_ativo

FROM luccagomes.midia m

WHERE m.id = :NEW.midia\_id;

IF v\_ativo = 'N' THEN

RAISE\_APPLICATION\_ERROR(

-20001,

'ERRO: Não é possível comprar mídia inativa (ID=' || :NEW.midia\_id || ').'

);

END IF;

END;

/

*PROMPT* ... Trigger luccagomes.trg\_check\_midia\_ativa\_compra criado com sucesso.

*-------------------------------------------------------------------------------*

*PROMPT* 3) INSERINDO DADOS BÁSICOS PARA TESTE

*-------------------------------------------------------------------------------*

*PROMPT* 3.1) Inserindo um usuário (ID gerado pelo IDENTITY):

*-- Se 'fulano\_login' ou 'fulano\_login2' já existirem, mude o valor do login novamente.*

INSERT INTO luccagomes.usuario (nome, login, senha, nasc)

VALUES ('fulano37', 'fulano\_login37', '1234', *DATE* '1990-01-01');

/

*PROMPT* 3.2) Inserindo uma mídia (ID gerado pelo IDENTITY):

INSERT INTO luccagomes.midia (titulo, dt\_lancamento, valor, duracao)

VALUES ('Filme Ativo', *DATE* '2020-01-01', 50.00, 120);

/

*PROMPT* 3.3) Criando uma nota\_fiscal para esse usuário (valor\_total=0):

*-- dt\_pagamento = SYSTIMESTAMP por default (PRIMARY KEY = (usuario\_id, dt\_pagamento))*

INSERT INTO luccagomes.nota\_fiscal (usuario\_id, valor\_total)

SELECT u.id, 0

FROM (

SELECT id

FROM luccagomes.usuario

ORDER BY id *DESC*

) u

WHERE ROWNUM = 1;

/

*PROMPT* 3.4) Consultar os dados recém inseridos:

SELECT \* FROM luccagomes.usuario;

SELECT \* FROM luccagomes.midia;

SELECT \* FROM luccagomes.nota\_fiscal;

*-------------------------------------------------------------------------------*

*PROMPT* 4) TESTES DOS TRIGGERS

*-------------------------------------------------------------------------------*

*PROMPT* 4.1) Tentando comprar com mídia ativa (deve funcionar):

INSERT INTO luccagomes.compra (usuario\_id, midia\_id, dt\_compra, valor)

SELECT

(SELECT id

FROM (SELECT id FROM luccagomes.usuario ORDER BY id *DESC*)

WHERE ROWNUM = 1

) AS usuario\_id,

(SELECT id

FROM (SELECT id FROM luccagomes.midia ORDER BY id *DESC*)

WHERE ROWNUM = 1

) AS midia\_id,

(SELECT dt\_pagamento

FROM (SELECT dt\_pagamento FROM luccagomes.nota\_fiscal ORDER BY dt\_pagamento *DESC*)

WHERE ROWNUM = 1

) AS dt\_compra,

NULL AS valor

FROM DUAL;

/

*PROMPT* Consulta para ver se o valor foi preenchido automaticamente:

SELECT \*

FROM luccagomes.compra

ORDER BY dt\_compra *DESC*;

*PROMPT* 4.2) Tornar *a* mídia inativa e tentar comprar de novo (deve falhar):

UPDATE luccagomes.midia

*SET* ativo = 'N'

WHERE id = (

SELECT id

FROM (SELECT id FROM luccagomes.midia ORDER BY id *DESC*)

WHERE ROWNUM = 1

);

/

*PROMPT* Ao tentar comprar agora, deve disparar o erro -20001:

INSERT INTO luccagomes.compra (usuario\_id, midia\_id, dt\_compra, valor)

SELECT

(SELECT id

FROM (SELECT id FROM luccagomes.usuario ORDER BY id *DESC*)

WHERE ROWNUM = 1

),

(SELECT id

FROM (SELECT id FROM luccagomes.midia ORDER BY id *DESC*)

WHERE ROWNUM = 1

),

(SELECT dt\_pagamento

FROM (SELECT dt\_pagamento FROM luccagomes.nota\_fiscal ORDER BY dt\_pagamento *DESC*)

WHERE ROWNUM = 1

),

NULL

FROM DUAL;

/

*PROMPT* ... Espera-se erro ORA-20001: ERRO: Não é possível comprar mídia inativa.

*PROMPT* ========== FIM DOS TRIGGERS E TESTES ==========

### **7.2 Relatório**

O trigger trg\_auto\_valor\_compra ajusta automaticamente o valor de uma compra caso ele não seja informado. Ele verifica o preço da mídia associada à compra e preenche o campo de valor com esse preço. Se a mídia tiver um valor definido, a compra será registrada com esse valor.

O trigger trg\_check\_midia\_ativa\_compra impede que compras sejam feitas para mídias inativas. Ele verifica se a mídia está ativa antes de permitir a compra. Caso a mídia esteja inativa, o sistema bloqueia a operação e exibe uma mensagem de erro.

Os testes realizados inserem um usuário, uma mídia ativa e uma nota fiscal. Em seguida, tentam registrar uma compra para a mídia ativa (que deve funcionar) e outra para a mesma mídia após torná-la inativa (que deve falhar).

### **7.3 Output.**

