CENTRO UNIVERSITÁRIO SATC
Engenharia da Computação – Disciplina: **Seg. em redes de computadores**
Professor: **Jorge Luiz da Silva**

RELATÓRIO FINAL: SISTEMA DE GESTÃO DE ACADEMIA (GymManager)

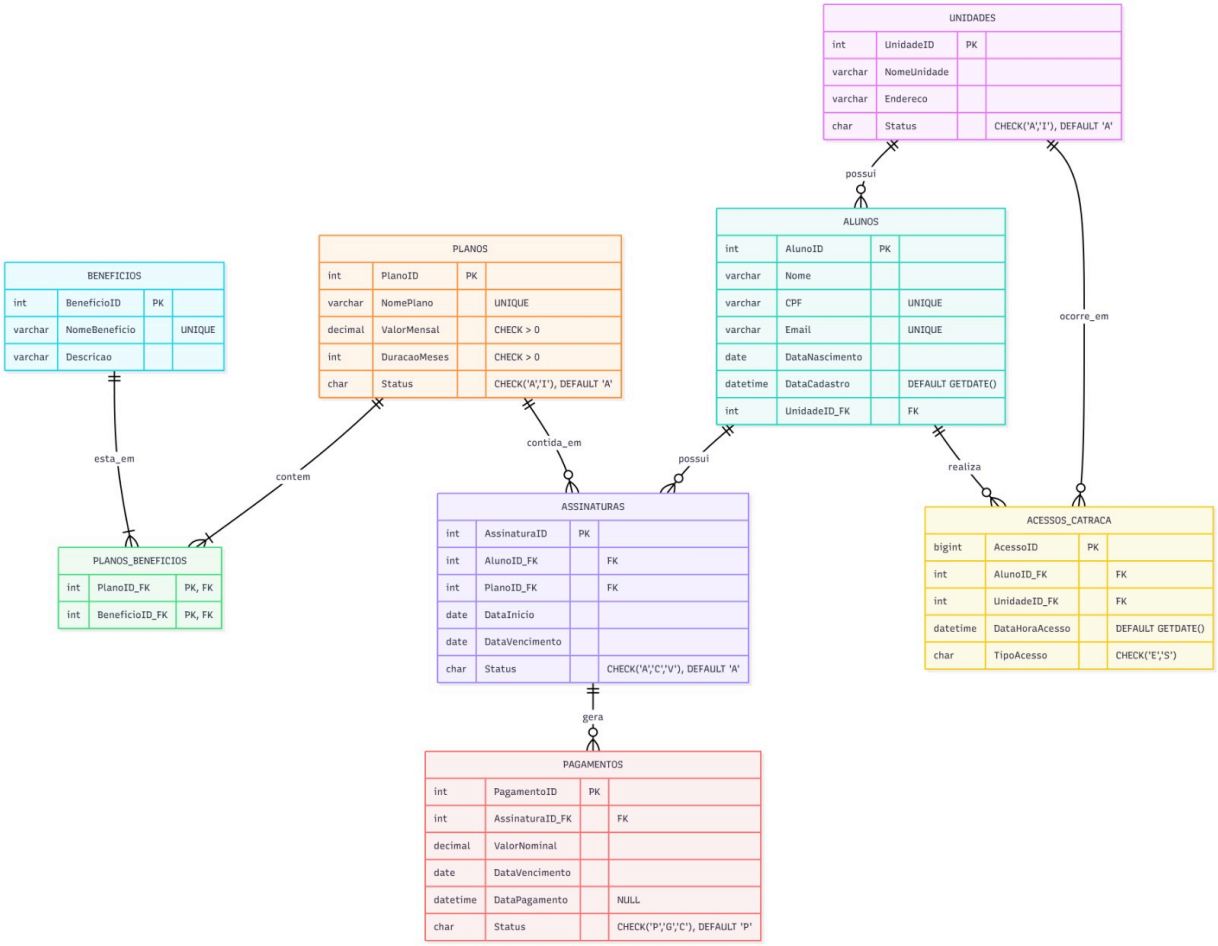
Disciplina: Banco de Dados II
Tema: Assinatura para Academias
Alunos:
João Victor da Rosa Falcão, Leticia Vociekoski, Luiz Coral e Thalysson Dos Santos .

1. INTRODUÇÃO E ESCOPO

O projeto GymManager tem como objetivo informatizar o controle de uma rede de academias. O sistema gerencia desde o cadastro de unidades e planos, passando pela matrícula de alunos, controle financeiro de mensalidades e validação de acesso físico (catraca).

2. MODELO FÍSICO DO BANCO DE DADOS

O modelo foi construído para atender à Terceira Forma Normal (3NF), garantindo integridade e evitando redundância.



Justificativa de Normalização:

1FN: Todos os atributos são atômicos e não existem grupos de repetição.

2FN: Todas as tabelas possuem Chave Primária (PK) e os atributos não-chave dependem totalmente da PK.

3FN: Não existem dependências transitivas. Exemplo: Os detalhes do plano (Valor, Duração) ficam na tabela PLANOS, e não repetidos na tabela ASSINATURAS.

3. DICIONÁRIO DE DADOS E OTIMIZAÇÃO

Para garantir performance e economia de armazenamento, foram aplicadas as seguintes otimizações nos tipos de dados:

- TINYINT: Utilizado na coluna DuracaoMeses da tabela PLANOS.
- CHAR(1): Utilizado para colunas de Status ('A','I','P','G') e Tipo ('E','S').
- DATE vs DATETIME: Usado de forma estratégica.

Dicionario de dados:

Tabela	Unidades				
Descrição	Armazena as filiais da academia				
Nome da Coluna	**Tipo de Dado**	**Tamanho**	**Constraints (PK, FK, etc)**	**Descrição**	
UnidadeID	INT	-	PK, IDENTITY(1,1), NOT NULL	Identificador único da unidade	
NomeUnidade	VARCHAR	100	NOT NULL	Nome da filial	
Endereco	VARCHAR	150	NULL	Endereço da unidade	
Status	CHAR	1	DEFAULT 'A'	A' = Ativa, 'I' = Inativa	
Índice					
Nome do índice	**Clustered**	**NonClustered**	**Unique**	**Colunas**	**Justificativa**
PK_UNIDADES	X		X	UnidadeID	Chave Primária (Padrão)

Tabela	Planos				
Descrição	Armazena os tipos de planos oferecidos				
Nome da Coluna	**Tipo de Dado**	**Tamanho**	**Constraints (PK, FK, etc)**	**Descrição**	
PlanoID	INT	-	PK, IDENTITY(1,1), NOT NULL	Identificador único do plano	
NomePlano	VARCHAR	50	NOT NULL	Nome do plano (ex: Mensal)	
ValorMensal	DECIMAL	10,2	NOT NULL	Valor da mensalidade	
DuracaoMeses	TINYINT	-	NOT NULL	Otimização: Duração em meses (0-255)	
Índice					
Nome do índice	**Clustered**	**NonClustered**	**Unique**	**Colunas**	**Justificativa**
PK_PLANOS	X		X	PlanoID	Chave Primária (Padrão)

Tabela	Beneficios				
Descrição	Armazena os benefícios/extras que um plano pode ter (ex: Acesso à Piscina)				
Nome da Coluna	**Tipo de Dado**	**Tamanho**	**Constraints (PK, FK, etc)**	**Descrição**	
BeneficioID	INT	-	PK, IDENTITY(1,1), NOT NULL	Identificador único do benefício	
NomeBeneficio	VARCHAR	100	NOT NULL	Nome curto do benefício	
Descricao	VARCHAR	200	NULL	Descrição detalhada	
Índice					
Nome do índice	**Clustered**	**NonClustered**	**Unique**	**Colunas**	**Justificativa**
PK_BENEFÍCIOS	X		X	BeneficioID	Chave Primária (Padrão)

Tabela	Alunos				
Descrição	Armazena os clientes/membros da academia				
Nome da Coluna	**Tipo de Dado**	**Tamanho**	**Constraints (PK, FK, etc)**	**Descrição**	
AlunoID	INT	-	PK, IDENTITY(1,1), NOT NULL	Identificador único do aluno	
Nome	VARCHAR	100	NOT NULL	Nome completo do aluno	
CPF	VARCHAR	11	UNIQUE, NOT NULL	CPF (apenas números)	
Email	VARCHAR	100	NULL	Email de contato	
DataNascimento	DATE	-	NOT NULL	Data de nascimento	
UnidadeID_FK	INT	-	FK (ref. Unidades), NOT NULL	Unidade de origem do aluno	
Índice					
Nome do índice	**Clustered**	**NonClustered**	**Unique**	**Colunas**	**Justificativa**
PK_ALUNOS	X		X	AlunoID	Chave Primária (Padrão)
UQ_ALUNOS_CPF		X	X	CPF	Garante unicidade do CPF
idx_alunos_unidade		X		UnidadeID_FK	Melhora performance nos JOINs com Unidades
idx_alunos_nascimento		X		DataNascimento	Otimiza a Pergunta de Negócio 3 (Média de Idade)

Tabela	PLANOS_BENEFICIOS				
Descrição	Armazena planos				
Nome da Coluna	**Tipo de Dado**	**Tamanho**	**Constraints (PK, FK, etc)**	**Descrição**	
PlanoID_FK	INT	-	PK, FK (ref. Planos)	Chave composta do plano	
BeneficioID_FK	INT	-	PK, FK (ref. Beneficios)	Chave composta do benefício	
Índice					
Nome do índice	**Clustered**	**NonClustered**	**Unique**	**Colunas**	**Justificativa**
PK_PLANOS_BENEFICIOS	X		X	PlanoID_FK, BeneficioID_FK	Chave Primária Composta

Tabela	Assinaturas				
Descrição	Tabela principal que liga Alunos e Planos (controla a vigência)				
Nome da Coluna	**Tipo de Dado**	**Tamanho**	**Constraints (PK, FK, etc)**	**Descrição**	
AssinaturaID	INT	-	PK, IDENTITY(1,1), NOT NULL	Identificador da assinatura	
AlunoID_FK	INT	-	FK (ref. Alunos)	Aluno titular	
PlanoID_FK	INT	-	FK (ref. Planos)	Plano contratado	
DataInicio	DATE	-	NOT NULL	Início da vigência	
DataVencimento	DATE	-	NOT NULL	Fim da vigência	
Índice					
Nome do índice	**Clustered**	**NonClustered**	**Unique**	**Colunas**	**Justificativa**
PK_ASSINATURAS	X		X	AssinaturaID	Chave Primária (Padrão)
idx_assinaturas_aluno		X		AlunoID_FK	Busca rápida do histórico do aluno
idx_assinaturas_plano_data		X		PlanoID_FK, DataInicio	Otimiza a Pergunta de Negócio 5 (Ranking de Vendas)

Tabela	Pagamentos				
Descrição	Armazena o histórico de parcelas de cada assinatura				
Nome da Coluna	**Tipo de Dado**	**Tamanho**	**Constraints (PK, FK, etc)**	**Descrição**	
PagamentoID	INT	-	PK, IDENTITY(1,1), NOT NULL	Identificador da parcela	
AssinaturaID_FK	INT	-	FK (ref. Assinaturas)	Assinatura vinculada	
ValorNominal	DECIMAL	10,2	NOT NULL	Valor original da parcela	
DataVencimento	DATE	-	NOT NULL	Data limite	
Status	CHAR	1	NOT NULL	P'=Pendente, 'G'=Grave, 'C'=Confirmado	
DataPagamento	DATETIME	-	NULL	Data real da baixa (NULL se não pago)	
Índice					
Nome do índice	**Clustered**	**NonClustered**	**Unique**	**Colunas**	**Justificativa**
PK_PAGAMENTOS	X		X	PagamentoID	Chave Primária (Padrão)
idx_pagamentos_assinatura		X		AssinaturaID_FK	Busca rápida para verificar inadimplência no Trigger
idx_pagamentos_status_venc		X		Status, DataVencimento (Include: ValorNominal)	Otimiza a Pergunta de Negócio 1 (Receita) e 4 (Devedores)

Tabela	Acessos_Catraca				
Descrição	Log de entradas (e saídas) dos alunos nas unidades				
Nome da Coluna	**Tipo de Dado**	**Tamanho**	**Constraints (PK, FK, etc)**	**Descrição**	
AcessoID	INT	-	PK, IDENTITY(1,1), NOT NULL	Log de acesso	
AlunoID_FK	INT	-	FK (ref. Alunos)	Aluno que acessou	
UnidadeID_FK	INT	-	FK (ref. Unidades)	Local do acesso	
DataHoraAcesso	DATETIME	-	DEFAULT GETDATE()	Carimbo de tempo exato	
TipoAcesso	CHAR	1	NOT NULL	E'=Entrada, 'S'=Saída	
Índice					
Nome do índice	**Clustered**	**NonClustered**	**Unique**	**Colunas**	**Justificativa**
PK_ACESSOS_CATRACA	X		X	AcessoID	Chave Primária (Padrão)
idx_catraca_aluno		X		AlunoID_FK	Verificação rápida se o aluno acessou
idx_acessos_datahora		X		DataHoraAcesso	Otimiza a Pergunta de Negócio 2 (Frequência) e 6 (Pico)

4. OBJETOS DE BANCO DE DADOS (MANDATÓRIOS)

A. Stored Procedure – sp_CadastrarAluno

```
CREATE PROCEDURE sp_CadastrarAluno
  @Nome VARCHAR(100), @CPF VARCHAR(11), @Email VARCHAR(100),
  @DataNascimento DATE, @UnidadeID INT
AS
BEGIN
  INSERT INTO ALUNOS (Nome, CPF, Email, DataNascimento, UnidadeID_FK)
  VALUES (@Nome, @CPF, @Email, @DataNascimento, @UnidadeID);
END;
```

B. Function – fn_CalcularIdade

```
CREATE FUNCTION fn_CalcularIdade(@DataNasc DATE)
RETURNS INT
AS
BEGIN
  RETURN DATEDIFF(YEAR, @DataNasc, GETDATE());
END;
```

C. Trigger – trg_BloquearAcesso

```
CREATE TRIGGER trg_BloquearAcesso ON ACESSOS_CATRACA AFTER INSERT AS
BEGIN
  IF EXISTS (
    SELECT 1 FROM PAGAMENTOS P
    JOIN ASSINATURAS A ON A.AssinaturaID = P.AssinaturaID_FK
    JOIN INSERTED I ON I.AlunoID_FK = A.AlunoID_FK
    WHERE P.Status = 'G'
  )
  BEGIN
    RAISERROR ('Acesso bloqueado: pagamento em atraso.', 16, 1);
    ROLLBACK TRANSACTION;
  END
END;
```

5. APLICAÇÃO CRUD (INTERFACE GRÁFICA)

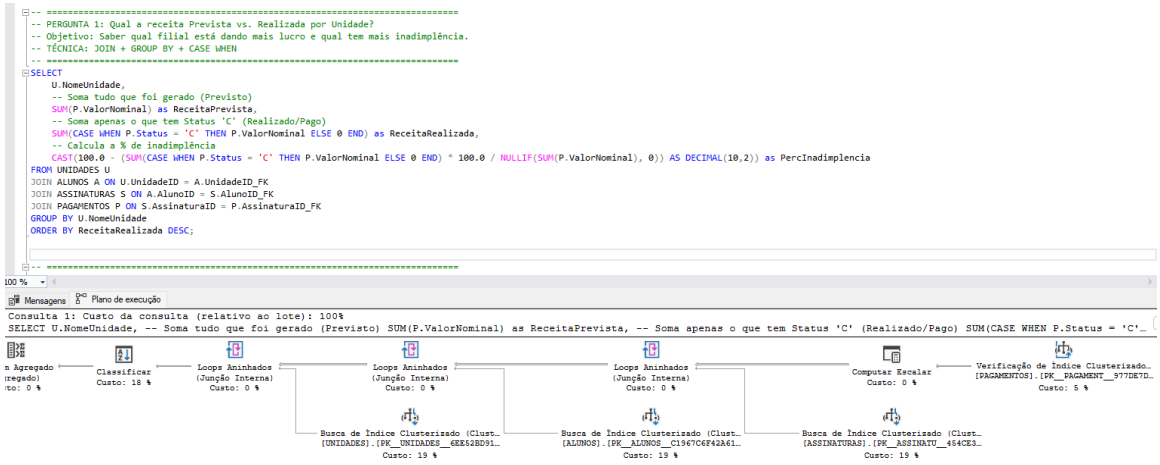
A aplicação foi desenvolvida em C# (Windows Forms) utilizando System.Data.SqlClient.

(Cole aqui o print das telas: Cadastro, Prontuário, Catraca)

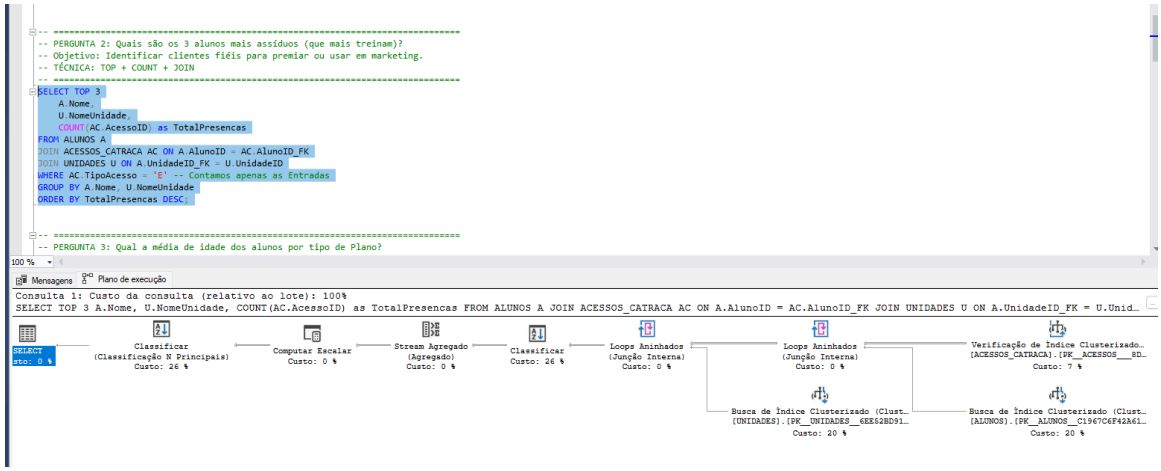
6. PERGUNTAS DE NEGÓCIO

Inclui consultas SQL estratégicas, índices utilizados e prints dos resultados.

Pergunta 1:

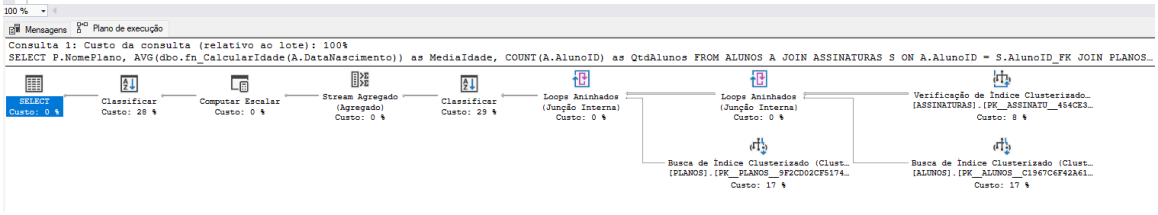


Pergunta 2:



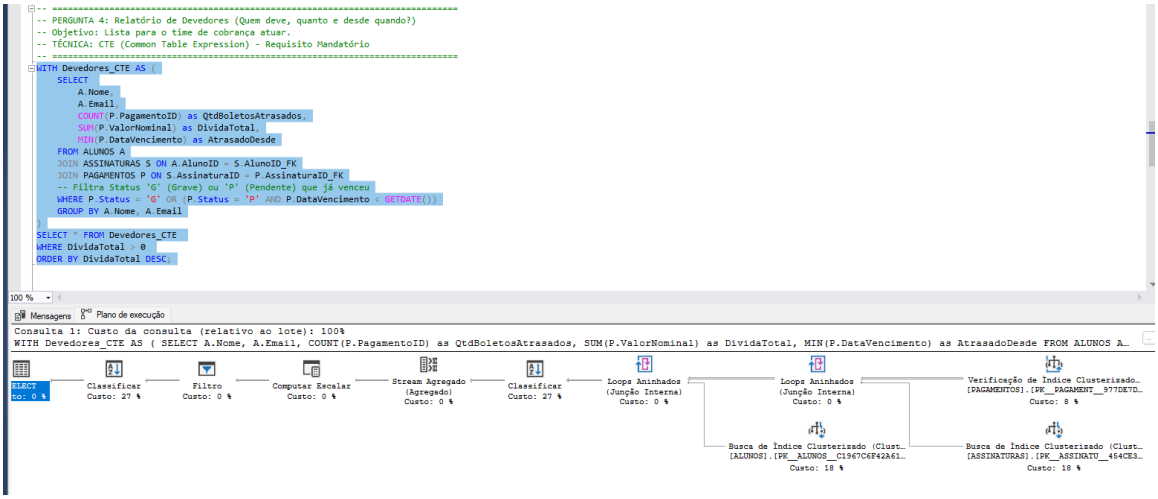
Pergunta 3:

```sql
-- ================================================================================
-- PERGUNTA 3: Qual a média de idade dos alunos por tipo de Plano?
-- Objetivo: Entender o perfil demográfico para direcionar campanhas (Ex: Plano Boxe é para jovens?).
-- TÉCNICA: FUNCTION (fn_CalcularIdade) + AVG
-- ================================================================================
SELECT
    P.NomePlano,
    AVG(dbo.fn_CalcularIdade(A.DataNascimento)) as MediaIdade,
    COUNT(A.AlunoID) as QtdAlunos
FROM ALUNOS A
JOIN ASSINATURAS S ON A.AlunoID = S.AlunoID_FK
JOIN PLANOS P ON S.PlanoID_FK = P.PlanoID
WHERE S.DataVencimento >= GETDATE() -- Apenas alunos com plano ativo
GROUP BY P.NomePlano
ORDER BY MediaIdade ASC;
```



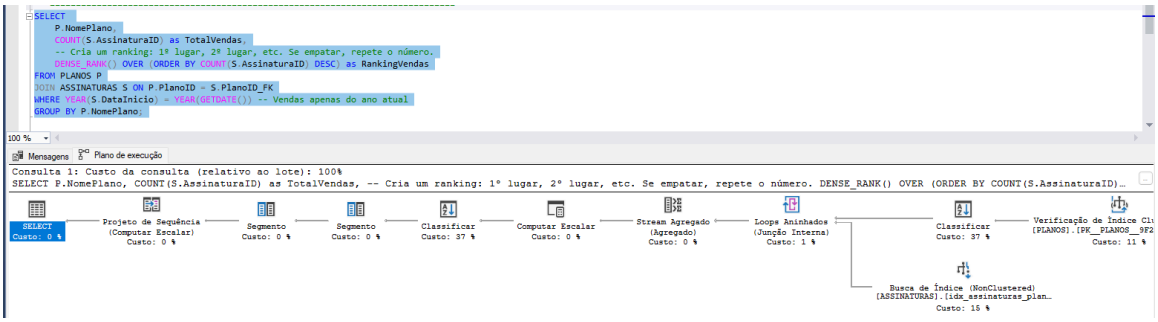
Pergunta 4:

```sql
-- ================================================================================
-- PERGUNTA 4: Relatório de Devedores (Quem deve, quanto e desde quando?)
-- Objetivo: Lista para o time de cobrança atuar.
-- TÉCNICA: CTE (Common Table Expression) - Requisito Mandatório
-- ================================================================================
WITH Devedores_CTE AS (
    SELECT
        A.Nome,
        A.Email,
        COUNT(P.PagamentoID) as QtdBoletosAtrasados,
        SUM(P.ValorNominal) as DividaTotal,
        MIN(P.DataVencimento) as AtrasadoDesde
    FROM ALUNOS A
    JOIN ASSINATURAS S ON A.AlunoID = S.AlunoID_FK
    JOIN PAGAMENTOS P ON S.AssinaturaID = P.AssinaturaID_FK
    -- Filtra Status 'G' (Grave) ou 'P' (Pendente) que já venceu
    WHERE P.Status = 'G' OR (P.Status = 'P' AND P.DataVencimento < GETDATE())
    GROUP BY A.Nome, A.Email
)
SELECT * FROM Devedores_CTE
WHERE DividaTotal > 0
ORDER BY DividaTotal DESC;
```



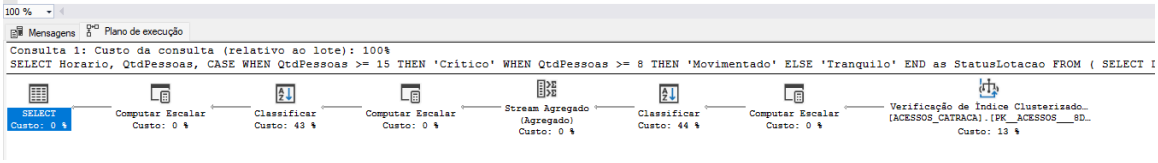
Pergunta 5:

```sql
-- ================================================================================
SELECT
    P.NomePlano,
    COUNT(S.AssinaturaID) as TotalVendas,
    -- Cria um ranking: 1º lugar, 2º lugar, etc. Se empatar, repete o número.
    DENSE_RANK() OVER (ORDER BY COUNT(S.AssinaturaID) DESC) as RankingVendas
FROM PLANOS P
JOIN ASSINATURAS S ON P.PlanoID = S.PlanoID_FK
WHERE YEAR(S.DataInicio) = YEAR(GETDATE()) -- Vendas apenas do ano atual
GROUP BY P.NomePlano;
```



Pergunta 6:

```sql
--
SELECT
    Horario,
    QtdPessoas,
    CASE
        WHEN QtdPessoas >= 15 THEN 'Crítico'
        WHEN QtdPessoas >= 8 THEN 'Movimentado'
        ELSE 'Tranquilo'
    END as StatusLotacao
FROM (
    SELECT
        DATEPART(HOUR, DataHoraAcesso) as Horario,
        COUNT(*) as QtdPessoas
    FROM ACESSOS_CATRACA
    WHERE TipoAcesso = 'E'
    GROUP BY DATEPART(HOUR, DataHoraAcesso)
) as TabelaVirtual
ORDER BY QtdPessoas DESC;
```



7. GITHUB

Link do repositório:
https://github.com/Luizcoral/Projeto-Final-Banco-de-Dados-2-.git