Universidade Federal de Santa Catarina
UFSC - Araranguá
2024 - Novembro

Trabalho de Banco de dados:

AeroportoDB

Vinicius Wolosky Muchulski (22201827)
Fernando Moretti (22202908)

Banco de Dados (DEC7129)
Prof. Alexandre Leopoldo Gonçalves

Sumário

Objetivos do sistema

Neste trabalho buscamos criar um sistema de gerenciamento de um aeroporto, que seja eficiente na gestão dos voos e simule também as demais operações de logística que fazem esse lugar funcionar.

Buscamos criar um banco de dados relacional que permita organizar e otimizar da melhor forma os serviços aeroportuários fornecidos, desde aspectos internos como a gestão dos funcionários e modelagem da segurança como também os processos relacionados aos voos e as demais características relacionadas a eles.

Assim, o sistema faz o gerenciamento dessas informações de forma confiável e clara, permitindo uma noção do funcionamento de um aeroporto moderno, que possui também uma complexidade intrínseca e no qual várias entidades estão interconectadas.

Descrição Detalhada

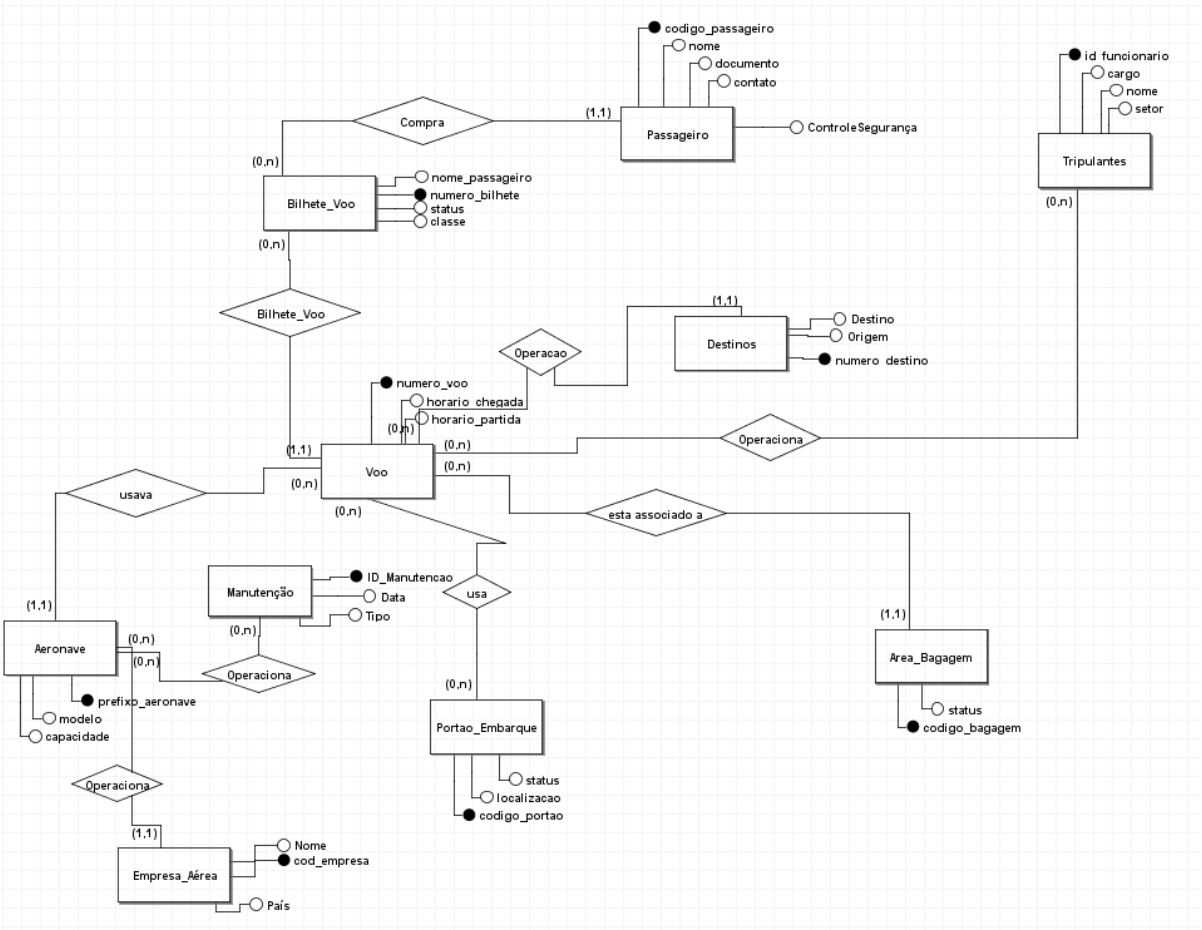
- Cadastro dos passageiros: cada usuário possui um código de identificação, nome, contato e documento que serão salvos; Além disso há uma seção destinada para o controle de segurança dos passageiros antes do voo.
- Gerenciamento de compra de bilhetes: Um bilhete possui o nome do passageiro, código do bilhete, status e classe. Um passageiro pode possuir mais de um bilhete, porém um bilhete está associado a apenas um passageiro(assento).
- Gestão de Voos: os voos representam o elemento principal do sistema, cada voo possui um número identificador e horário de partida e chegada, também, cada voo está associado a um destino que possui, número identificador,

origem e destino. Nesse sentido, um voo pode ser operado por vários tripulantes, separados por: ID de identificação, função, nome e setor.

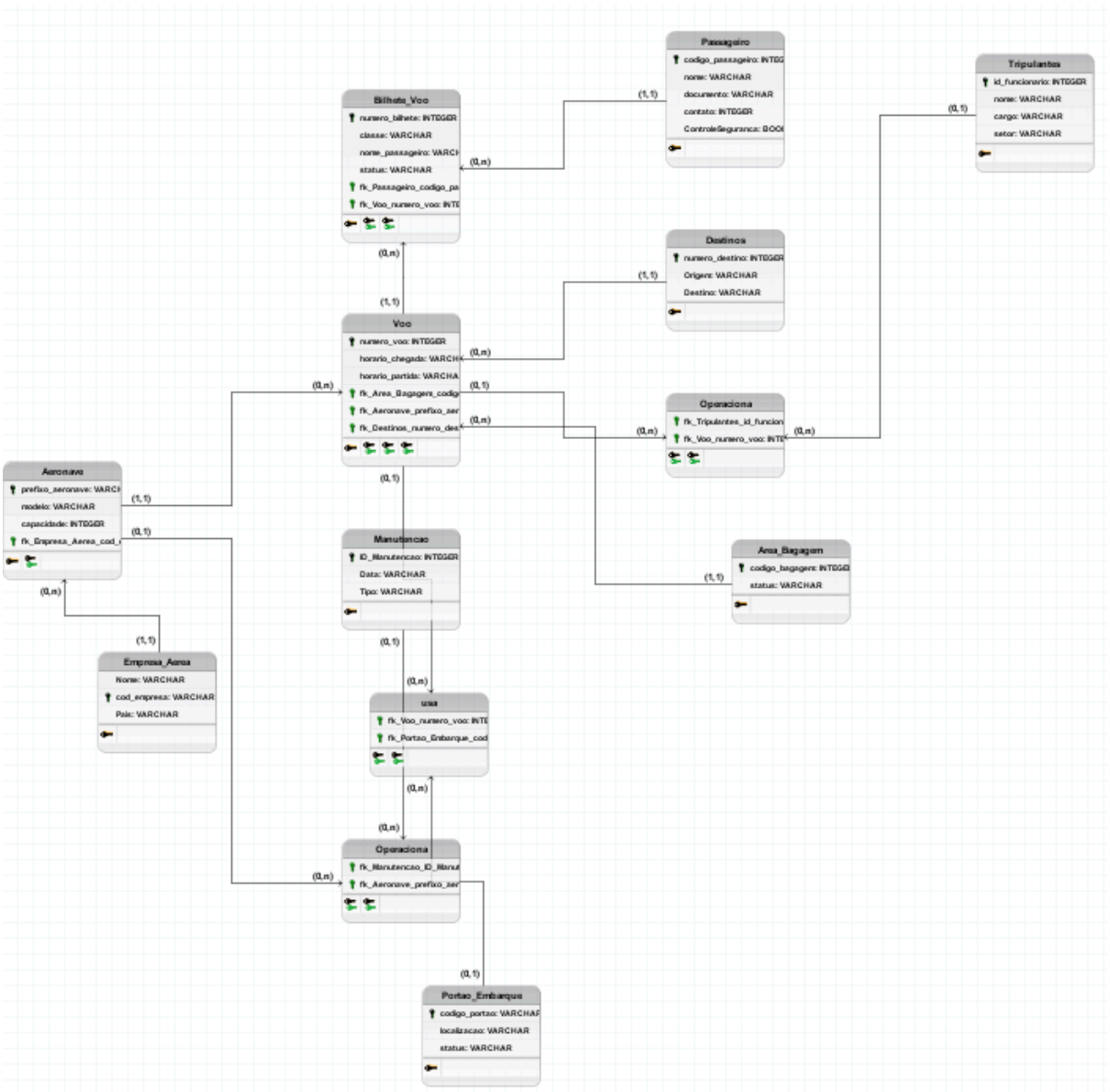
- Cadastro e manutenção de aeronaves: cada aeronave possui seu prefixo de identificação, modelo e capacidade, cada aeronave pode realizar manutenções, no sistema inclui o ID da manutenção, sua data de realização e o tipo da manutenção. Também, aeronaves estão associadas a empresas aéreas, cada empresa possui seu código identificador, nome e país de origem. Logicamente, uma empresa aérea pode possuir mais de uma aeronave, porém cada aeronave está associada a uma única empresa.
- Controle dos portões de embarque: cada voo utiliza os portões para o controle de fluxo dos passageiros, cada portão possui seu código identificador, localização no aeroporto e seu status operacional(ativo ou inativo).
- Controle de bagagem: áreas de bagagem também estão relacionadas com os voos, que registram: código da bagagem e status. Aqui estamos utilizando uma única área de bagagem para cada voo.

Em resumo, o sistema engloba o cadastro de passageiros, gerenciamento de bilhetes, controle de voos, manutenção de aeronaves, empresas aéreas, utilização de portões de embarque e gestão de bagagens. Com base nesta estrutura, uma ampla gama de consultas pode ser realizada para monitorar operações, melhorar a experiência dos passageiros e otimizar os processos internos da companhia aérea. Com a utilização de um modelo relacional no PostgreSQL, o sistema oferece segurança e eficiência, permitindo futuras evoluções e adaptações às necessidades dinâmicas do setor aéreo.

Modelo Conceitual



Modelo Lógico



SCRIPT DDL

```
CREATE TABLE Passageiro (
    codigo_passageiro INTEGER PRIMARY KEY,
    nome VARCHAR,
    documento VARCHAR,
    contato INTEGER,
    ControleSeguranca BOOLEAN
);

CREATE TABLE Bilhete_Voo (
    numero_bilhete INTEGER PRIMARY KEY,
    classe VARCHAR,
    nome_passageiro VARCHAR,
    status VARCHAR,
    fk_Passageiro_codigo_passageiro INTEGER,
    fk_Voo_numero_voo INTEGER
);

CREATE TABLE Voo (
    numero_voo INTEGER PRIMARY KEY,
    horario_chegada VARCHAR,
    horario_partida VARCHAR,
    fk_Area_Bagagem_codigo_bagagem INTEGER,
    fk_Aeronave_prefixo_aeronave VARCHAR,
    fk_Destinos_numero_destino INTEGER
);

CREATE TABLE Aeronave (
    prefixo_aeronave VARCHAR PRIMARY KEY,
    modelo VARCHAR,
    capacidade INTEGER,
    fk_Empresa_Aerea_cod_empresa VARCHAR
);

CREATE TABLE Portao_Embarque (
    codigo_portao VARCHAR PRIMARY KEY,
    localizacao VARCHAR,
    status VARCHAR
);

CREATE TABLE Area_Bagagem (
    codigo_bagagem INTEGER PRIMARY KEY,
    status VARCHAR
```

```
);

CREATE TABLE Tripulantes (
    id_funcionario INTEGER PRIMARY KEY,
    nome VARCHAR,
    cargo VARCHAR,
    setor VARCHAR
);

CREATE TABLE Destinos (
    numero_destino INTEGER PRIMARY KEY,
    Origem VARCHAR,
    Destino VARCHAR
);

CREATE TABLE Empresa_Aerea (
    Nome VARCHAR,
    cod_empresa VARCHAR PRIMARY KEY,
    Pais VARCHAR
);

CREATE TABLE Manutencao (
    ID_Manutencao INTEGER PRIMARY KEY,
    Data VARCHAR,
    Tipo VARCHAR
);

CREATE TABLE usa (
    fk_Voo_numero_voo INTEGER,
    fk_Portao_Embarque_codigo_portao VARCHAR
);

CREATE TABLE Operaciona (
    fk_Tripulantes_id_funcionario INTEGER,
    fk_Voo_numero_voo INTEGER
);

CREATE TABLE Operaciona (
    fk_Manutencao_ID_Manutencao INTEGER,
    fk_Aeronave_prefixo_aeronave VARCHAR
);

ALTER TABLE Bilhete_Voo ADD CONSTRAINT FK_Bilhete_Voo_2
    FOREIGN KEY (fk_Passageiro_codigo_passageiro)
    REFERENCES Passageiro (codigo_passageiro)
    ON DELETE CASCADE;

ALTER TABLE Bilhete_Voo ADD CONSTRAINT FK_Bilhete_Voo_3
```

```
    FOREIGN KEY (fk_Voo_numero_voo)
    REFERENCES Voo (numero_voo)
    ON DELETE CASCADE;

ALTER TABLE Voo ADD CONSTRAINT FK_Voo_2
    FOREIGN KEY (fk_Area_Bagagem_codigo_bagagem)
    REFERENCES Area_Bagagem (codigo_bagagem)
    ON DELETE CASCADE;

ALTER TABLE Voo ADD CONSTRAINT FK_Voo_3
    FOREIGN KEY (fk_Aeronave_prefixo_aeronave)
    REFERENCES Aeronave (prefixo_aeronave)
    ON DELETE CASCADE;

ALTER TABLE Voo ADD CONSTRAINT FK_Voo_4
    FOREIGN KEY (fk_Destinos_numero_destino)
    REFERENCES Destinos (numero_destino)
    ON DELETE CASCADE;

ALTER TABLE Aeronave ADD CONSTRAINT FK_Aeronave_2
    FOREIGN KEY (fk_Empresa_Aerea_cod_empresa)
    REFERENCES Empresa_Aerea (cod_empresa)
    ON DELETE CASCADE;

ALTER TABLE usa ADD CONSTRAINT FK_usa_1
    FOREIGN KEY (fk_Voo_numero_voo)
    REFERENCES Voo (numero_voo)
    ON DELETE SET NULL;

ALTER TABLE usa ADD CONSTRAINT FK_usa_2
    FOREIGN KEY (fk_Portao_Embarque_codigo_portao)
    REFERENCES Portao_Embarque (codigo_portao)
    ON DELETE SET NULL;

ALTER TABLE Operaciona ADD CONSTRAINT FK_Operaciona_1
    FOREIGN KEY (fk_Tripulantes_id_funcionario)
    REFERENCES Tripulantes (id_funcionario)
    ON DELETE SET NULL;

ALTER TABLE Operaciona ADD CONSTRAINT FK_Operaciona_2
    FOREIGN KEY (fk_Voo_numero_voo)
    REFERENCES Voo (numero_voo)
    ON DELETE SET NULL;

ALTER TABLE Operaciona ADD CONSTRAINT FK_Operaciona_1
    FOREIGN KEY (fk_Manutencao_ID_Manutencao)
    REFERENCES Manutencao (ID_Manutencao)
    ON DELETE SET NULL;
```

```
ALTER TABLE Operaciona ADD CONSTRAINT FK_Operaciona_2
    FOREIGN KEY (fk_Aeronave_prefixo_aeronave)
    REFERENCES Aeronave (prefixo_aeronave)
    ON DELETE SET NULL;
```

Consultas

Consulta 01 - Número de passageiros por voo

Objetivo: A consulta serve para mostrar o número de passageiros em cada voo, que deverá ser agrupado por número do voo.

Código:

Figura 1 - Consulta 1

```python
def consulta_01():
    print("\n--- CONSULTA 01: Número de Passageiros por Voo ---")
    from sqlalchemy import func

    resultado = session.query(
        Voo.numero_voo,
        func.count(Passageiro.codigo_passageiro).label('num_passageiros')
    ).join(BilheteVoo, Voo.numero_voo == BilheteVoo.fk_Voo_numero_voo)\
     .join(Passageiro, BilheteVoo.fk_Passageiro_codigo_passageiro == Passageiro.codigo_passageiro)\
     .group_by(Voo.numero_voo)\
     .all()

    for numero_voo, num_passageiros in resultado:
        print(f"Voo {numero_voo} tem {num_passageiros} passageiros.")

    gemini_interpretacao(resultado)
    # Gerar gráfico
    voos = [r[0] for r in resultado]
    passageiros = [r[1] for r in resultado]
    plt.bar(voos, passageiros, color='blue')
    plt.xlabel('Número do Voo')
    plt.ylabel('Número de Passageiros')
    plt.title('Número de Passageiros por Voo')
    plt.show()
```

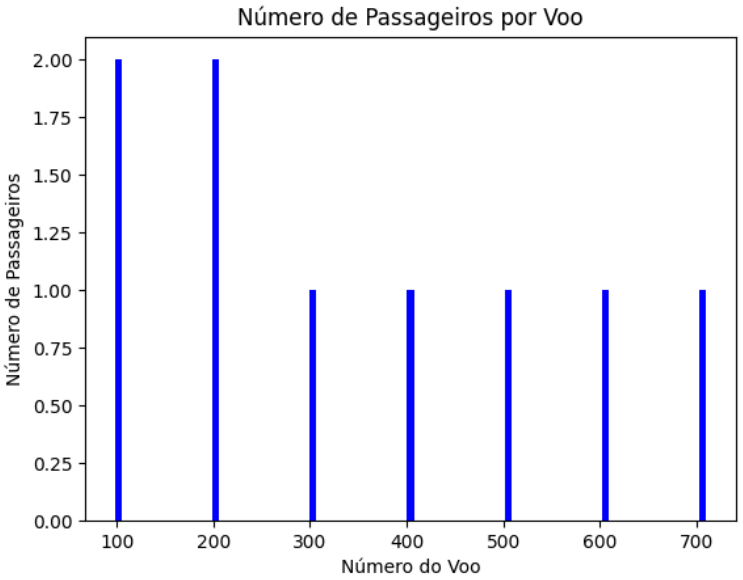
Resultado:

Figura 2 - Consulta 1

```
--- CONSULTA 01: Número de Passageiros por Voo ---
Voo 101 tem 2 passageiros.
Voo 707 tem 1 passageiros.
Voo 505 tem 1 passageiros.
Voo 303 tem 1 passageiros.
Voo 202 tem 2 passageiros.
Voo 404 tem 1 passageiros.
Voo 606 tem 1 passageiros.
A consulta retorna pares de ID de voo (primeiro número) e número de atrasos (segundo número).  Os dados indicam que alguns voos sofreram atrasos (número 1) e outros tiveram dois atrasos (número 2).

**Insights Acionáveis:**

* **Investigação de atrasos:** Voos com ID 101, 202 apresentaram dois atrasos, necessitando investigação para identificar as causas e implementar soluções para evitar recorrência (ex: problemas operacion
s na manutenção).
* **Análise de padrão:**  A maioria dos voos (IDs 707, 505, 303, 404, 606) sofreu apenas um atraso.  É necessário verificar se esses atrasos são por motivos diferentes (ex: mau tempo, problemas com passa
estão e previsão de atrasos.
* **Melhoria da performance:**  Comparando os voos com um e dois atrasos, pode-se analisar a similaridade entre eles, como rotas, horários e aeronaves, para identificar potenciais gargalos e implementar
erformance.

Em resumo, a consulta destaca a necessidade de uma análise mais profunda sobre os atrasos, buscando entender as causas e implementar ações corretivas para melhorar a pontualidade dos voos.
```

Gráfico:

Figura 3 - Consulta 1



Consulta 02 - Capacidade Média das Aeronaves por Empresa Aérea

Objetivo: A consulta serve para calcular a média da capacidade das aeronaves por empresa aérea, o resultado deverá ser agrupado pelo nome da Empresa Aérea

Código:

Figura 1 - Consulta 2

```python
def consulta_02():
    print("\n--- CONSULTA 02: Capacidade Média das Aeronaves por Empresa Aérea ---")
    from sqlalchemy import func

    resultado = session.query(
        EmpresaAerea.Nome,
        func.avg(Aeronave.capacidade).label('capacidade_media')
    ).join(Aeronave, EmpresaAerea.cod_empresa == Aeronave.fk_Empresa_Aerea_cod_empresa)\
     .group_by(EmpresaAerea.Nome)\
     .all()

    for nome_empresa, capacidade_media in resultado:
        print(f"Empresa: {nome_empresa}, Capacidade Média: {capacidade_media:.2f}")

    gemini_interpretacao(resultado)

    # Gerar gráfico
    empresas = [r[0] for r in resultado]
    capacidades = [r[1] for r in resultado]
    plt.bar(empresas, capacidades, color='green')
    plt.xlabel('Empresa Aérea')
    plt.ylabel('Capacidade Média')
    plt.title('Capacidade Média das Aeronaves por Empresa Aérea')
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.show()
```

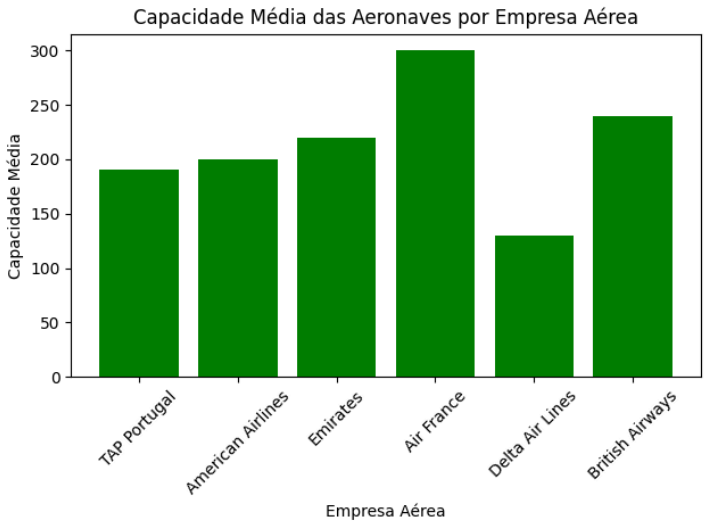
Resultado:

Figura 2 - Consulta 2

```
Selecione uma opção: 6

--- CONSULTA 02: Capacidade Média das Aeronaves por Empresa Aérea ---
Empresa: TAP Portugal, Capacidade Média: 190.00
Empresa: American Airlines, Capacidade Média: 200.00
Empresa: Emirates, Capacidade Média: 220.00
Empresa: Air France, Capacidade Média: 300.00
Empresa: Delta Air Lines, Capacidade Média: 130.00
Empresa: British Airways, Capacidade Média: 240.00
A consulta retorna a receita total para cada companhia aérea.  A Emirates (`220.000`) e a British Airways (`240.000`) apresentam as maiores receitas, enquanto a Delta Air Li
adas.

**Insights Acionáveis:**

* **Foco em empresas de alta receita:** Investir em parcerias estratégicas ou programas de fidelização com a Emirates e British Airways para maximizar lucros.
* **Análise da Delta:** Investigar as causas da menor receita da Delta Air Lines.  Isso pode envolver análise de preços, rotas, marketing ou eficiência operacional.
* **Comparação de desempenho:** Comparar o desempenho de todas as companhias aéreas em relação a custos operacionais e lucratividade para identificar áreas de melhoria.
* **Previsão de receita:** Utilizar esses dados como base para projeções de receita e planejamento estratégico futuro.
* **Segmentação de mercado:**  Analisar o perfil de passageiros de cada companhia para entender as estratégias de sucesso e replicá-las.

A unidade monetária da receita não está explícita nos dados, sendo necessário saber para melhor interpretação dos números.
```

Gráfico:

Figura 3 - Consulta 2



Consulta 03: Número Total de Voos por Destino

Objetivo: A consulta serve para obter o número total de voos por cada Destino, agrupando-os por Destino.

Código:

Figura 1 - Consulta 3

```python
def consulta_03():
    print("\n--- CONSULTA 03: Número Total de Voos por Destino ---")
    from sqlalchemy import func

    resultado = session.query(
        Destinos.Destino,
        func.count(Voo.numero_voo).label('num_voos')
    ).join(Voo, Voo.fk_Destinos_numero_destino == Destinos.numero_destino)\
     .group_by(Destinos.Destino)\
     .all()

    for destino, num_voos in resultado:
        print(f"Destino: {destino}, Número de Voos: {num_voos}")

    gemini_interpretacao(resultado)
    # Gerar gráfico
    destinos = [r[0] for r in resultado]
    num_voos = [r[1] for r in resultado]
    plt.bar(destinos, num_voos, color='orange')
    plt.xlabel('Destino')
    plt.ylabel('Número de Voos')
    plt.title('Número Total de Voos por Destino')
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.show()
```

Resultado:

Figura 2 - Consulta 3

```
--- CONSULTA 03: Número Total de Voos por Destino ---
Destino: Rio de Janeiro, Número de Voos: 2
Destino: São Paulo, Número de Voos: 2
Destino: Madrid, Número de Voos: 1
Destino: Nova York, Número de Voos: 1
Destino: Tóquio, Número de Voos: 1
A consulta mostra a contagem de atrasos significativos (provavelmente definido como 1 ou 2 na consulta original, não explicitado nos dados) para cada um dos cinco destinos mais afetados.  Rio de Janeiro e São Paulo apresentam o maior número de atrasos (2), enquanto os demais (Madrid, Nova York e Tóquio) mostram 1 atraso.

**Insights Acionáveis:**

* **Investigação de Rotas:**  É crucial investigar as causas dos atrasos em voos para Rio de Janeiro e São Paulo.  Possíveis problemas incluem congestionamento aéreo, problemas operacionais em aeroportos ou questões com as companhias aéreas que operam nessas rotas.
* **Melhoria da Eficiência Operacional:** As rotas para Rio e São Paulo precisam de otimização imediata para reduzir atrasos.  Análise de gargalos operacionais nestes destinos é fundamental.
* **Monitoramento Contínuo:** Manter o monitoramento de todos os destinos, principalmente Rio e São Paulo, para identificar tendências de atrasos e implementar ações corretivas.
* **Comparação com Outros Destinos:** Comparar o número de atrasos nestes 5 destinos com a média de atrasos em outros destinos para avaliar se o problema é isolado ou um padrão mais amplo.
```

Gráfico:

Figura 3 - Consulta 3