

Fase tre

Indice degli argomenti

- [Indice degli argomenti](#)
- [Progetto VRoomA](#)
 - [Componenti del gruppo](#)
 - [Motivazioni](#)
 - [Obiettivi](#)
 - [Raccolta dei dati](#)
 - [Analisi dei requisiti](#)
 - [Glossario delle entità](#)
 - [Glossario dei termini](#)
 - [Glossario delle relazioni](#)
 - [Schemi](#)
 - [Schema Scheletro](#)
 - [Raffinazione](#)
 - [Schema Concettuale](#)
 - [Normalizzazione](#)
 - [Generalizzazione](#)
 - [Schema Finale](#)
 - [Schema Logico](#)
 - [Schema Fisico](#)
 - [Implementazione Database - MySQL](#)
 - [Creazione delle tabelle](#)
 - [Triggers](#)
 - [Stored Procedures](#)
 - [Inserimenti](#)
 - [Script di creazione automatica di query](#)
 - [Query](#)
 - [Ottimizzazione](#)
 - [Algebra Relazionale](#)
 - [Calcolo Relazionale](#)
 - [Sicurezza](#)
 - [Views](#)
 - [Creazione Utenti](#)

Progetto VRoomA

Componenti del gruppo

Nome	Cognome	Matricola	Mail
Leonardo	Ascenzi	0310858	leonardo.ascenzi@students.uniroma2.eu
Franco	Salvucci	0306604	franco.salvucci@students.uniroma2.eu
Nicolò	Spadoni	0311175	nicolo.spadoni@students.uniroma2.eu

Motivazioni

Il database che stiamo realizzando è incentrato all'implementazione di un software dedicato all'organizzazione di viaggi tramite taxi.

Obiettivi

L'obiettivo principale di questo sistema è permettere agli utenti di organizzare gli spostamenti tramite taxi a seconda delle proprie esigenze, del tipo di veicolo scelto e del costo della tratta scelta.

Da un punto di vista societario, gli obiettivi sono quelli di valutare la qualità del lavoro degli autisti tramite i feedback forniti dai clienti e migliorare dove possibile il servizio.

Raccolta dei dati

Analisi dei requisiti

I ruoli aziendali sono i seguenti:

- Autisti
- Manutentori

Gli **autisti** potranno scegliere se accettare o rifiutare la corsa, specificando in questo caso la motivazione del rifiuto.

Inoltre potranno lasciare un **feedback** all'utente riguardo il comportamento prima e durante la corsa.

Ogni **autista** ha la propria macchina privata, e può contattare i manutentori aziendali in caso di guasto del veicolo.

Ad ogni **autista** sono assegnati uno o più **turni** di lavoro, con il vincolo che il singolo autista non può essere assegnato a due turni lavorativi che hanno orario inizio e orario fine uguali.

I **manutentori** possono ricevere richieste di assistenza da parte degli autisti e contattare le officine convenzionate per effettuare il lavoro di assistenza.

Le **officine** non fanno parte della società.

Le tipologie di **veicolo** disponibili sono le seguenti:

- Base: 4 posti disponibili
- Plus: 7 posti disponibili, adibito a trasporto di carrozzine per disabili
- Premium: 12 posti disponibili, adibito a trasporto di carrozzine per disabili

Ogni **veicolo**, identificato in modo univoco dalla targa, per poter circolare, deve essere assicurato.

Quando si prenota una **corsa (tratta)** si possono scegliere due punti:

- Partenza, ovvero la **via** da dove si vuole iniziare la corsa
- Arrivo, ovvero la **via** in cui termina la corsa

Ogni **prenotazione** può essere accettata o rifiutata in base a determinate esigenze dell'**utente** (Es: prenotazione effettuata per errore) e dell'**autista** (Es: indisponibilità al servizio).

Ad ogni tratta completata è associato un feedback che può essere lasciato sia dall'**utente** che dall'**autista**.

Ogni **utente** può effettuare illimitate richieste di **prenotazione**, in base alle necessità personali (numero di passeggeri, persone con disabilità, punto di ritiro, punto di rilascio), con il vincolo di una corsa per volta.

A corsa completata l'**utente** può lasciare un **feedback** con un numero di stelle (da 1 a 5) e un commento.

Ogni **utente** deve aggiungere una **carta** con cui effettuare il pagamento relativo alla tratta effettuata, in un secondo momento potrà aggiungere altri metodi di pagamento secondo le proprie esigenze.

Ogni **utente** può aggiungere alla lista dei preferiti una qualunque delle tratte effettuate da lui, scegliendo se aggiungere solo la tratta o anche l'autista.

Ogni **utente** può accedere alla cronologia delle prenotazioni effettuate.

Glossario delle entità

Entità	Descrizione	Attributi	Relazioni Coinvolte
Patenti	Describe tutte le info riguardanti la patente degli autisti	Numero Patente , DDS, Categoria	Autisti
Manutentori	Addetti alla manutenzione delle auto degli autisti	ID_Manutentore , Email, BNumeroTelefono, DDN, Congome, Nome, Qualifica	Autisti
Autisti	Personale che svolge il ruolo di autista delle auto nella società	ID_Autista , Stipendio, Email, BNumeroTelefono, DDN, Congome, Nome	Patenti, Manutentori, Veicoli, TratteCompletate, TratteRifiutate, Turni
Veicoli	Auto utilizzate per il servizio di taxi	Targa , Marca, Modello, NumeroPosti	Autisti, Assicurazione
Turni	Turni lavorativi che riguardano gli autisti	OraInizio , OraFine	Autisti
Richiesta Prenotazione	Richieste di prenotazioni effettuate da parte dall'utente	Data , Ora , ID_Utente , Partenza , Arrivo , Numero Passeggeri	Utenti, Tratte Completate, Tratte Rifiutate, Fermate
Utenti	Utenti utilizzatori del servizio taxi	ID_Utente , Nome, Cognome, Email, Password	Carta, Richiesta Prenotazione, Feedback, Tratte completate
Feedback	Recensioni lasciate dall'utente e dagli autisti	ID_Feedback , StelleUtente, CommentoUtente, StelleAutista, CommentoAutista	Tratte Completate, Utenti, Autisti
Tratte Completate	Corse effettuate portate a termine con successo	Costo, MetodoDiPagamento	Richiesta Prenotazione, Feedback, Autisti
Tratte Rifiutate	Corse rifiutate da parte dell'autista per determinati motivi	Motivazione	Richiesta Prenotazione, Autisti
Carta	Carta di credito personale dell'utente	Numero Carta	Utenti
Assicurazioni	Dati dell'assicurazione associata al singolo veicolo	ID_Assicurazione , Data di scadenza, Tipo	Veicoli
Fermate	Elenco delle fermate disponibili	Latitudine , Longitudine , NomeFermata	Richiesta Prenotazione

Glossario dei termini

Entità	Descrizione	Sinonimi
Patente	Describe tutte le info riguardanti la patente degli autisti	Licenza di Guida
Manutentori	Addetti alla manutenzione delle auto degli autisti	Meccanici, Operai
Autisti	Personale che svolge il ruolo di autista delle auto nella società	Driver
Veicoli	Auto utilizzate per il servizio di taxi	Automobili
Turni	Turni lavorativi che riguardano gli autisti	Orario Lavorativo
Richiesta Prenotazione	Richieste di prenotazioni effettuate da parte dall'utente	Prenotazioni
Utenti	Utenti utilizzatori del servizio taxi	Personne
Feedback	Recensioni lasciate dall'utente e dagli autisti	Recensioni
Tratte Completate	Corse effettuate portate a termine con successo	Corse
Tratte Rifiutate	Corse rifiutate da parte dell'autista per determinati motivi	Corse Annulate

Entità	Descrizione	Sinonimi
Carta	Carta di credito personale dell'utente	Metodo di pagamento
Assicurazioni	Dati dell'assicurazione associata al singolo veicolo	RCA, Polizza assicurativa
Fermate	Elenco delle fermate selezionabili da un utente	Punti

Glossario delle relazioni

Relazione	Descrizione	Entità
VeicoloPossiedeAssicurazione	Relazione che dice che ogni veicolo ha una propria assicurazione	Veicoli (1,1), Assicurazione (1,1)
AutistaGuidaVeicolo	Relazione che dice che ogni autista guida la propria autovettura	Autisti (1,1), Veicoli (1,1)
AutistaPossiedePatente	Relazione che dice che ogni autista, per poter guidare, necessita di una patente	Autisti (1,1), Patente (1,1)
ContattaPerGuasto	Relazione che dice che ogni autista può (non necessariamente) contattare un manutentore per un guasto al veicolo	Autisti (0,N), Manutentori (0,N)
AutistaAccettaCorsa	Relazione che dice che ogni autista accetta la richiesta di prenotazione	Autisti (1,N), TratteCompletate (1,1)
AutistaRifiutaCorsa	Relazione che dice che l'autista rifiuta la corsa, in base a determinate circostanze	Autisti (1,N), TratteRifiutate (1,1)
UtentePossiedeCarta	Relazione che dice che ogni utente deve possedere almeno una carta con cui effettuare i pagamenti	Utenti (1,N), Carta (1,1)
EffettuaPrenotazione	Relazione che dice che ogni utente effettua una o più prenotazioni	Utenti (1,N), Richiesta Prenotazioni (1,1)
TrattaAvereFeedback	Relazione che dice che ogni tratta completata può avere (non necessariamente) un solo feedback, che viene lasciato dagli utenti e dagli autisti	Tratte Completate (0,1), Feedback (1,1)
TabellaOrarioLavorativo	Ogni autista ha un proprio turno lavorativo, ad ogni turno lavorativo vengono assegnati uno o più autisti	Autisti (1,N), Turni (1,N)
SceglierePartenza	Punto di partenza scelto per il ritiro da parte dell'autista	Fermate(1,N), RichiestaPrenotazioni(1,1)
ScegliArrivo	Punto di fine scelto per il rilascio da parte dell'autista	Fermate(1,N), RichiestaPrenotazioni(1,1)
Accettare	L'autista accetta la tratta	Autisti(0,N), TratteCompletate(1,1)
Rifiutare	L'autista rifiuta la tratta	Autisti(0,N), TratteRifiutate(1,1)

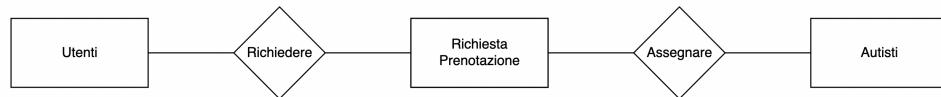
Schemi

Schema Scheletro

Le entità principali del sistema sono le seguenti:

- *Utenti*
- *RichiestaPrenotazioni*
- *Autisti*

La relazione che intercorre tra queste entità ci permette di affermare che l'*Utente* può effettuare una *Prenotazione* che verrà poi assegnata ad un singolo *Autista*.

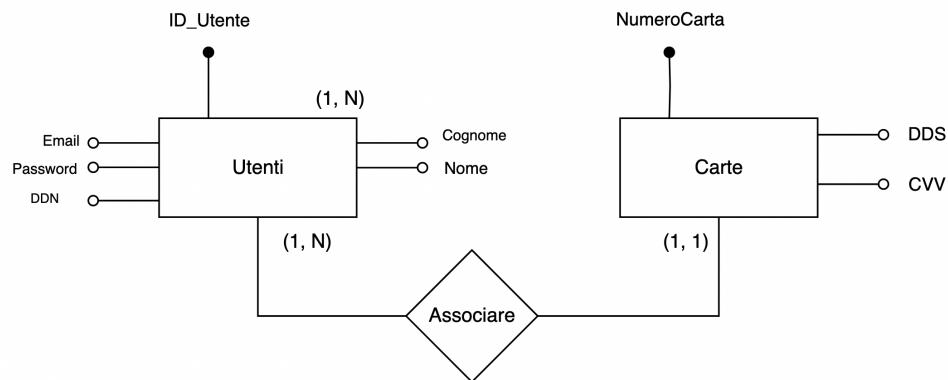


Raffinazione

In questo caso stiamo raffinando l'entità *Utenti*.

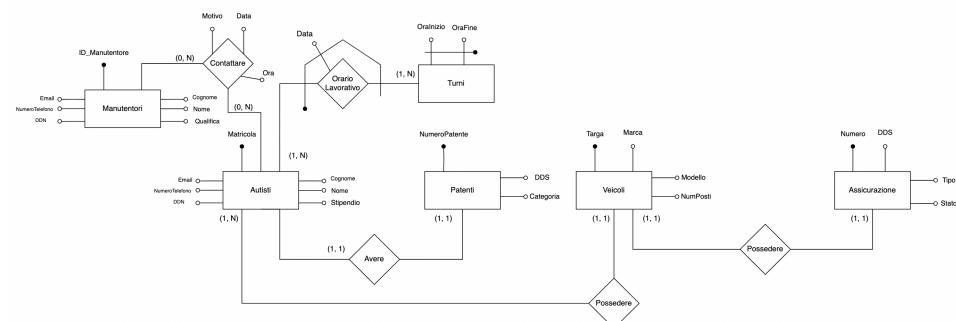
- Ad ogni *Utente* è associata una o più *Carte* con cui poi l'*Utente* effettuerà i vari pagamenti.

I dati della *Carta* non sono salvati nel database per questioni di privacy, bensì, verranno prelevati tramite interrogazioni al database della banca (che non fa parte del nostro sistema)



In questo caso stiamo raffinando l'entità *Autisti*.

- La prima relazione che si ha descrive il comportamento tra *Autisti* e *Turni*, un *Autista* ha uno o più *Turni* lavorativi, con il vincolo che non può avere due *Turni* uguali. Ad un *Turno* sono assegnati uno o più *Autisti*
- La seconda relazione che si ha descrive il comportamento tra *Autisti* e *Patenti*, un *Autista* possiede una ed una sola *Patente*. Una *Patente* è posseduta da uno ed un solo *Autista*
- La terza relazione che si ha descrive il comportamento tra *Autisti* e *Veicoli*, un *Autista* possiede uno più *Veicoli* con cui effettua le sue corse. Il *Veicolo* è privato e quindi è associato ad un singolo *Autista*.
- La quarta relazione che si ha descrive il comportamento tra gli *Autisti* e i *Manutentori*, un *Autista* può contattare uno o più *Manutentori*. Un *Manutentore* può essere contattato da uno o più *Autisti*
- L'ultima relazione descrive il comportamento tra *Veicoli* e *Assicurazioni*, ovvero, ad ogni *Veicolo* è associata una sola *Assicurazione*. Viceversa per le *Assicurazioni*.

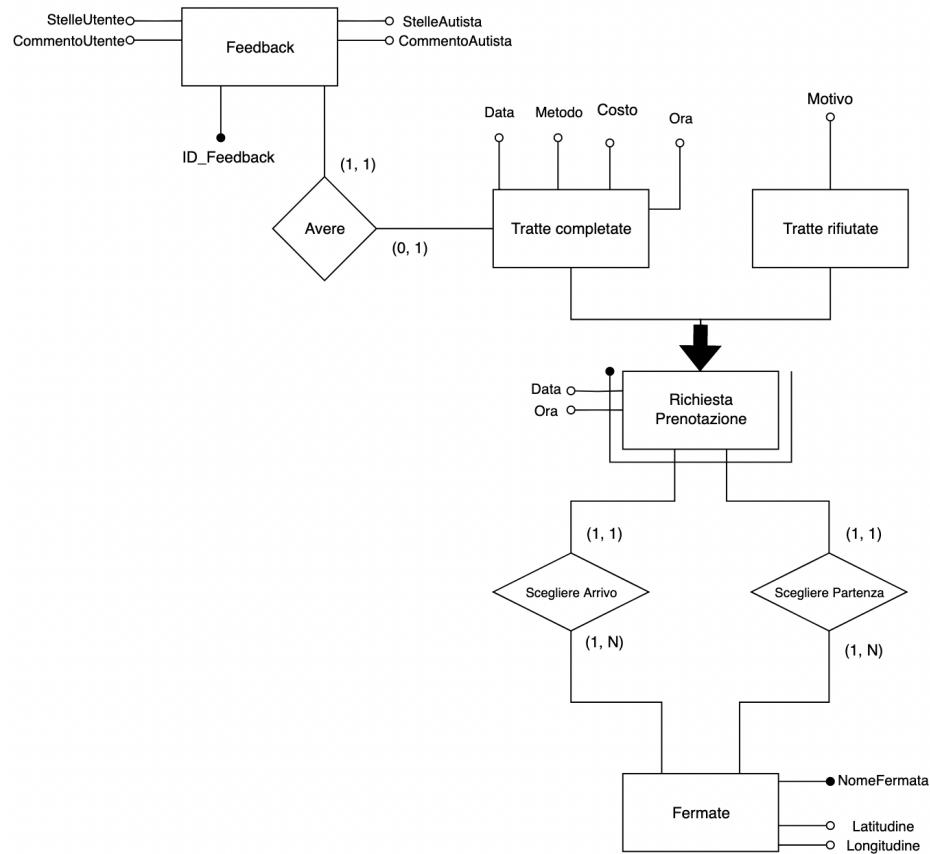


In questo caso stiamo raffinando l'entità *RichiestaPrenotazione*.

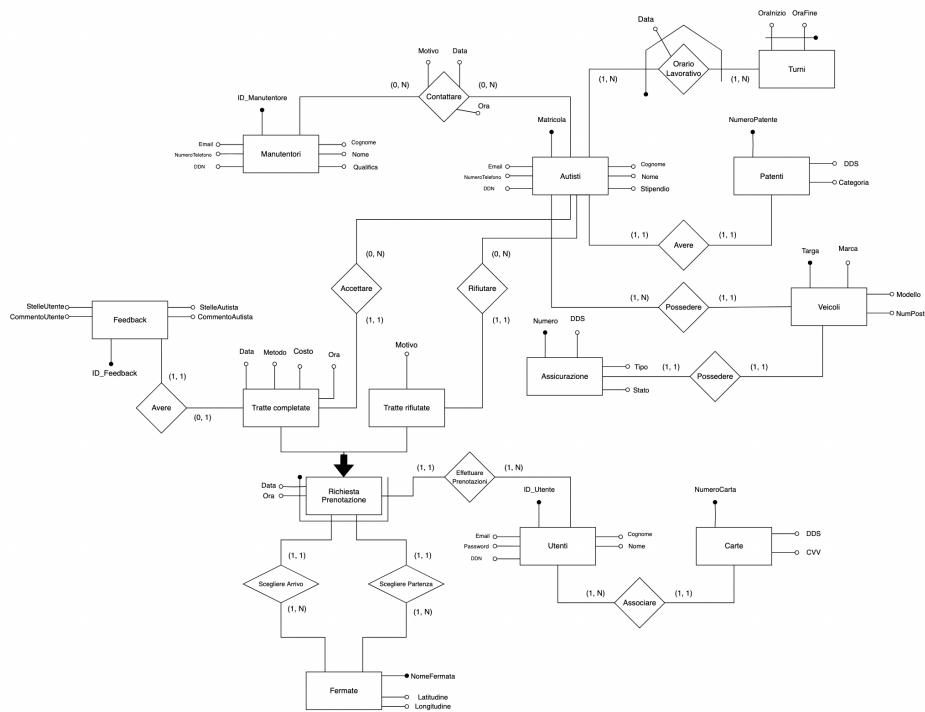
- La prima relazione che si ha permette alla *Prenotazione* di decidere uno ed un solo punto di partenza tra le *Fermate*. Le *Fermate* hanno più di un punto di partenza.
- La seconda relazione che si ha permette alla *Prenotazione* di decidere uno ed un solo punto di arrivo tra le *Fermate*. Le *Fermate* hanno più di un punto di arrivo.

Successivamente si evidenzia il fatto che la *RichiestaPrenotazione* ha 2 entità figlie (*TratteCompletate*, *TratteRifiutate*) che verranno generalizzate più avanti.

- Infine troviamo l'ultima relazione che descrive il comportamento tra le *Tratte completate* e *Feedback*. Una *Tratta Completata* può avere uno ed un solo *Feedback*. Un *Feedback* appartiene ad una sola *Tratta Completata*.



Schema Concettuale



Normalizzazione

Di seguito si discutono le forme normali dello schema concettuale:

- **1NF**: tutti gli schemi di relazione nello schema logico sopra riportato sono in **1NF**, poiché tutti gli attributi sono semplici, ovvero contengono soltanto valori atomici indivisibili.
- **2NF**: tutti gli schemi di relazione dello schema logico sono anche già in **2NF**, poiché sono già in **1NF** e nessun attributo presenta alcuna dipendenza parziale. Tutti gli attributi dipendono funzionalmente solo dalla chiave primaria della stessa tabella.
- **3NF**: tutti gli schemi di relazione sono anche in **3NF** perché già in **2NF**, ed inoltre, tutti gli attributi delle tabelle dipendono funzionalmente e direttamente dalla chiave primaria, senza transitività.

Generalizzazione

Una generalizzazione rappresenta un legame logico tra un'entità genitore e una o più entità figlie, in questo caso l'entità genitore è "Richiesta Prenotazione", e le entità figlie sono:

1. Richiesta prenotazione

1. Tratte completate
2. Tratte rifiutate

Abbiamo tre metodi per rappresentare una generalizzazione a livello fisico:

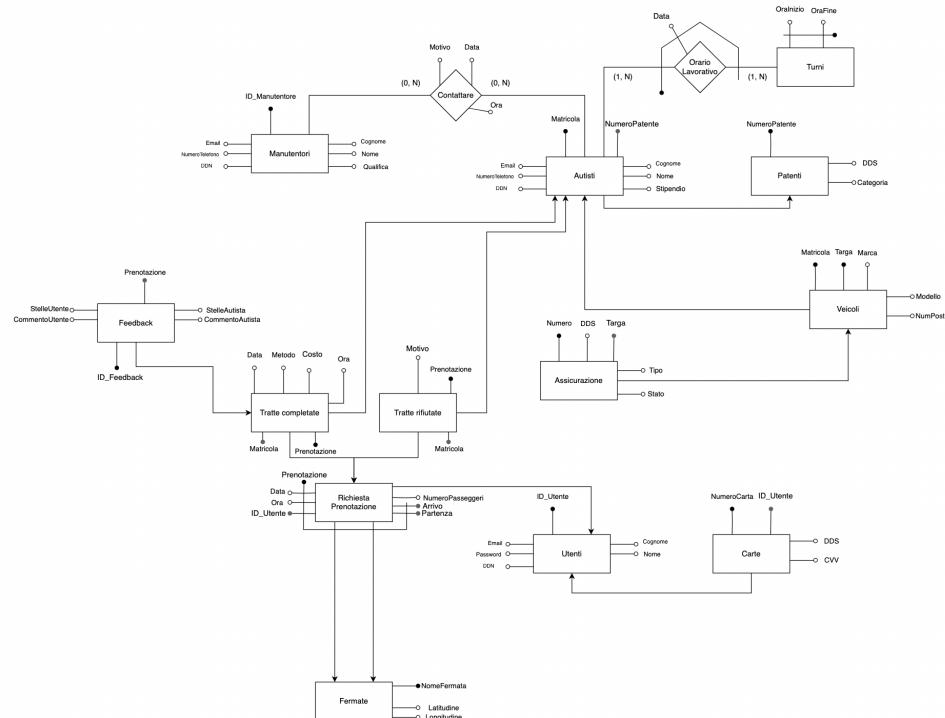
- Accorpamento del padre nelle entità figlie
- Accorpamento delle entità figlie nel padre
- Sostituzione della generalizzazione con relazioni

Tra questi metodi abbiamo scelto il terzo in quanto da noi considerato il più adeguato. Infatti, il primo metodo avrebbe portato ad una ridondanza di relazioni.

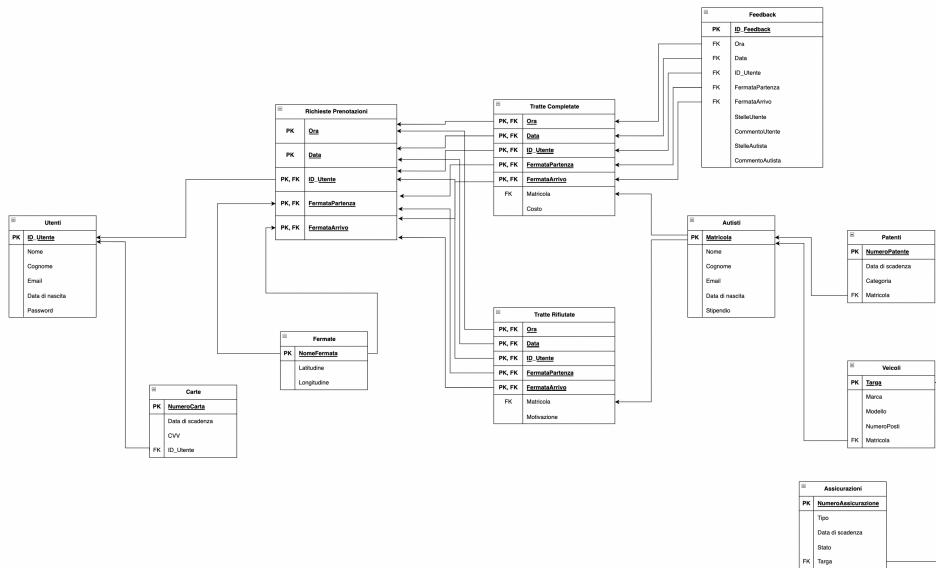
Il secondo metodo necessita dell'aggiunta di un attributo nell' entità "Richiesta Prenotazioni", ovvero il tipo di prenotazione (Es. Completata = 1 e Rifiutata = 2), in più si sarebbe dovuto scegliere se perdere informazioni (attributi) dei figli o inserire le informazioni nel padre, quindi aggiungere attributi dei figli al padre. La seconda scelta avrebbe portato ad una quantità non indifferente di valori NULL.

Nelle entità, le chiavi secondarie sono identificate con il pallino grigio, mentre quelle primarie sono identificate con il pallino nero.

Schema Finale



Schema Logico

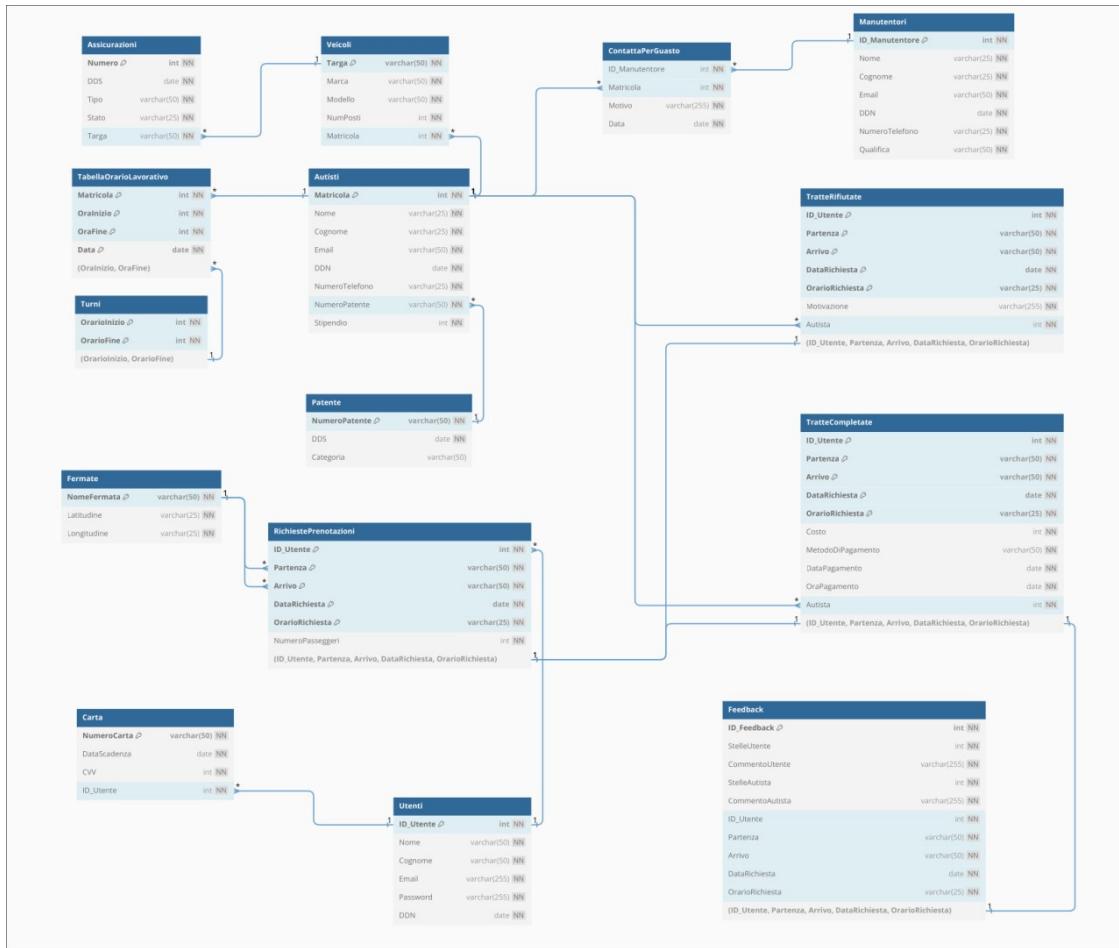


Le chiavi primarie sono identificate in **grassetto**, mentre le chiavi secondarie (o esterne) sono scritte in stile *Italic*

- Autisti (**Matricola**,Nome,Cognome,Email,DDN,*NumeroPatente*,Stipendio)
- Manutentori (**ID_Manutentore**,Nome,Cognome,Email,DDN,*NumeroTelefono*,Qualifica)
- ContattaPerGuasto (*ID_Manutentore*, **ID_Autista**,Motivo,Data)
- Turni (**OrarioInizio**, **OrarioFine**)
- TabellaOrarioLavorativo (**Matricola**,**OrarioInizio**,**OrarioFine**,Data)
- Veicoli (**Targa**, Marca, Modello, NumPosti,*Matricola*)
- Assicurazione (**Numero**, DataDiScadenza, Tipo,Stato,*Targa*)
- Utenti (**ID_Utente**, Nome, Cognome, Email, DDN, Password)
- Carta (**NumeroCarta**,DDS,CVV, *ID_Utente*)
- Richiesta Prenotazione (*ID_Utente*,*Partenza*,*Arrivo*,*DataRichiesta*,*OrarioRichiesta*, NumeroPasseggeri)
- Tratte Completate (*ID_Utente*,*Partenza*,*Arrivo*,*DataRichiesta*,*OrarioRichiesta*, Costo, MetodoDiPagamento, DataPagamento, OraPagamento, Matricola)
- Tratte Rifiutate (*ID_Utente*,*Partenza*,*Arrivo*,*DataRichiesta*,*OrarioRichiesta*, Motivazione, Matricola)
- Feedback (**ID_Feedback**, StelleUtente, CommentoUtente,StelleAutista, CommentoAutista,*ID_Utente*,*Partenza*,*Arrivo*,*DataRichiesta*,*OrarioRichiesta*)

Schema Fisico

Lo schema fisico è il seguente



Implementazione Database - MySQL

Creazione delle tabelle

```
use VroomA;

CREATE TABLE Patente (
    NumeroPatente varchar(50) not null,
    DDS date not null,
    Categoria varchar(50),
    PRIMARY KEY (NumeroPatente)
);
```

```
CREATE TABLE Manutentori (
    ID_Manutentore int not null ,
    Nome varchar(25) not null,
    Cognome varchar(25) not null,
    Email varchar(50) not null,
    DDS date not null,
    NumeroTelefono varchar(25) not null,
    Qualifica varchar(50) not null,
    PRIMARY KEY (ID_Manutentore)
```

```
CREATE TABLE Autisti (
    Matricola int not null ,
    Nome varchar(25) not null,
    Cognome varchar(25) not null,
    Email varchar(50) not null,
    DDS date not null,
    NumeroTelefono varchar(25) not null,
    NumeroPatente varchar(50) not null,
    Stipendio int not null,
```

```

        PRIMARY KEY (Matricola),
        FOREIGN KEY (NumeroPatente) REFERENCES Patente(NumeroPatente)
);

CREATE TABLE Veicoli (
    Targa varchar(50) not null,
    Marca varchar(50) not null,
    Modello varchar(50) not null,
    NumPosti int not null,
    Matricola int not null,
    PRIMARY KEY (Targa),
    FOREIGN KEY (Matricola) REFERENCES Autisti(Matricola)
);

CREATE TABLE Assicurazioni (
    Numero int not null,
    DDS date not null,
    Tipo varchar(50) not null,
    Stato varchar(25) not null,
    Targa varchar(50) not null,
    PRIMARY KEY (Numero),
    FOREIGN KEY (Targa) REFERENCES Veicoli(Targa)
);

CREATE TABLE Turni (
    OrarioInizio int not null,
    OrarioFine int not null,
    PRIMARY KEY (OrarioInizio,OrarioFine)
);

CREATE TABLE Utenti (
    ID_Utente int not null ,
    Nome varchar(50) not null,
    Cognome varchar(50) not null,
    Email varchar(255) not null,
    Password varchar(255) not null,
    DDN date not null,
    PRIMARY KEY (ID_Utente)
);

CREATE TABLE Fermate (
    NomeFermata varchar(50) not null,
    Latitudine varchar(25) not null,
    Longitudine varchar(25) not null,
    PRIMARY KEY (NomeFermata)
);

CREATE TABLE RichiestePrenotazioni (
    ID_Utente int not null,
    Partenza varchar(50) not null,
    Arrivo varchar(50) not null,
    DataRichiesta date not null,
    OrarioRichiesta varchar(25) not null,
    NumeroPasseggeri int not null,
    PRIMARY KEY (ID_Utente,Partenza,Arrivo,DataRichiesta,OrarioRichiesta),
    FOREIGN KEY (ID_Utente) REFERENCES Utenti(ID_Utente),
    FOREIGN KEY (Partenza) REFERENCES Fermate(NomeFermata),
    FOREIGN KEY (Arrivo) REFERENCES Fermate(NomeFermata)
);

```

```

CREATE TABLE Carta (
    NumeroCarta varchar(50) not null,
    DataScadenza date not null,
    CVV int not null,
    ID_Utente int not null,
    PRIMARY KEY (NumeroCarta),
    FOREIGN KEY (ID_Utente) REFERENCES Utenti(ID_Utente)
);

CREATE TABLE TratteCompletate (
    ID_Utente int not null,
    Partenza varchar(50) not null,
    Arrivo varchar(50) not null,
    DataRichiesta date not null,
    OrarioRichiesta varchar(25) not null,
    Costo int not null,
    MetodoDiPagamento varchar(50) not null,
    DataPagamento date not null,
    OraPagamento varchar(10) not null,
    Autista int not null,
    PRIMARY KEY (ID_Utente,Partenza,Arrivo,DataRichiesta,OrarioRichiesta),
    FOREIGN KEY (ID_Utente,Partenza,Arrivo,DataRichiesta,OrarioRichiesta)
    REFERENCES RichiestePrenotazioni(ID_Utente,Partenza,Arrivo,DataRichiesta,OrarioRichiesta),
    FOREIGN KEY (Autista) REFERENCES Autisti(Matricola)
);

CREATE TABLE TratteRifiutate (
    ID_Utente int not null,
    Partenza varchar(50) not null,
    Arrivo varchar(50) not null,
    DataRichiesta date not null,
    OrarioRichiesta varchar(25) not null,
    Motivazione varchar(255) not null,
    Autista int not null,
    PRIMARY KEY (ID_Utente,Partenza,Arrivo,DataRichiesta,OrarioRichiesta),
    FOREIGN KEY (ID_Utente,Partenza,Arrivo,DataRichiesta,OrarioRichiesta)
    REFERENCES RichiestePrenotazioni(ID_Utente,Partenza,Arrivo,DataRichiesta,OrarioRichiesta),
    FOREIGN KEY (Autista) REFERENCES Autisti(Matricola)
);

CREATE TABLE Feedback (
    ID_Feedback int not null ,
    StelleUtente int not null,
    CommentoUtente varchar(255) not null,
    StelleAutista int not null,
    CommentoAutista varchar(255) not null,
    ID_Utente int not null,
    Partenza varchar(50) not null,
    Arrivo varchar(50) not null,
    DataRichiesta date not null,
    OrarioRichiesta varchar(25) not null,
    PRIMARY KEY (ID_Feedback),
    FOREIGN KEY (ID_Utente,Partenza,Arrivo,DataRichiesta,OrarioRichiesta)
    REFERENCES TratteCompletate(ID_Utente,Partenza,Arrivo,DataRichiesta,OrarioRichiesta)
);

CREATE TABLE ContattaPerGuasto (
    ID_Manutentore int not null,
    Matricola int not null,
    Motivo varchar(255) not null,

```

```

        Data date not null,
        FOREIGN KEY (ID_Manutentore) REFERENCES Manutentori (ID_Manutentore),
        FOREIGN KEY (Matricola) REFERENCES Autisti (Matricola)
    );

CREATE TABLE TabellaOrarioLavorativo(
    Matricola int not null,
    OraInizio int not null,
    OraFine int not null,
    Data date not null,
    PRIMARY KEY(Matricola,OraInizio, OraFine,Data),
    FOREIGN KEY      (Matricola) REFERENCES Autisti(Matricola),
    FOREIGN KEY      (OraInizio,OraFine) REFERENCES Turni(OrarioInizio,OrarioFine)
);

```

Triggers

I trigger fanno parte del DDL (Data Definition Language), essi seguono il principio ECA, ovvero Event-Condition-Action. Solitamente, un trigger si può attivare prima o dopo un inserimento e hanno 2 livelli di granularità:

1. attivarsi per ogni tupla
2. attivarsi per ogni istruzione DML

Nello specifico in MySQL i trigger operano a livello di riga e si ammette un solo trigger per tabella. Osserviamo inoltre che questi vengono usati per mantenere constraint di ogni tipo, in primis il vincolo di integrità referenziale. Quelli di seguito sono una serie di trigger di esempio necessari per mantenere una serie di vincoli nel nostro database

Controlla Partenza e Arrivo

```

use VroomA;

DELIMITER //

CREATE TRIGGER `CheckPartenzaArrivo` BEFORE INSERT ON `RichiestePrenotazioni` FOR EACH ROW
BEGIN

-- Controlla se la partenza e l'arrivo sono uguali
    IF EXISTS (
        SELECT 1
        FROM RichiestePrenotazioni
        WHERE NEW.Partenza = NEW.Arrivo
    ) THEN

        -- Se lo sono, interrompi l'inserimento
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = '[ERRORE], LE FERMATE SCELTE NON POSSONO ESSERE UGUALI';
    END IF;

END //
DELIMITER

```

Stored Procedures

Le stored procedures sono un insieme di istruzioni SQL precompilate e memorizzate nella base di dati.

I vantaggi sono:

- **Efficienza:** Minimizzano il traffico di rete eseguendo le operazioni direttamente sul server
- **Sicurezza:** Limitano l'accesso diretto alle tabelle, controllando le operazioni consentite

- **Riutilizzabilità:** Possono essere richiamate da più punti dell'applicazione

Di seguito sono riportate alcune stored procedures che abbiamo creato per eseguire operazioni specifiche su due tabelle, ovvero **Utenti** e **Carte**

Maschera CVV

```
DELIMITER //

CREATE PROCEDURE MaskCVV()
BEGIN

    DECLARE done INT DEFAULT 0;
    DECLARE original_value VARCHAR(255);
    DECLARE str_length INT;
    DECLARE masked_value VARCHAR(255);
    DECLARE cur CURSOR FOR SELECT CVV FROM Carta;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

    OPEN cur;
    read_loop: LOOP
        FETCH cur INTO original_value;
        IF done THEN
            LEAVE read_loop;
        END IF;

        SET str_length = CHAR_LENGTH(original_value);
        SET masked_value = REPEAT('*', str_length);

        UPDATE Carta SET CVV = masked_value WHERE CVV = original_value;
    END LOOP;
    CLOSE cur;
END//


DELIMITER ;
```

Maschera Password Utente

```
DELIMITER //

CREATE PROCEDURE MaskPSW()
BEGIN

    DECLARE done INT DEFAULT 0;
    DECLARE original_value VARCHAR(255);
    DECLARE str_length INT;
    DECLARE masked_value VARCHAR(255);
    DECLARE cur CURSOR FOR SELECT Password FROM Utenti;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

    OPEN cur;
    read_loop: LOOP
        FETCH cur INTO original_value;
        IF done THEN
            LEAVE read_loop;
        END IF;

        SET str_length = CHAR_LENGTH(original_value);
        SET masked_value = REPEAT('*', str_length);

        UPDATE Utenti SET Password = masked_value WHERE Password = original_value;
    END LOOP;
END//
```

```
END LOOP;
CLOSE cur;
END//
```

```
DELIMITER ;
```

Inserimenti

Di seguito vengono riportati alcuni estratti di query per l'inserimento, presi dallo script di creazione automatica delle query

Patente

```
INSERT INTO Patenti (NumeroPatente,DDS,Categoria) VALUES ('VZMX360RW','2034-02-19','B96'),
('X78P0G5NM','2028-04-05','BE'),
('TE9KHF73D','2030-10-13','B'),
('PMOKYCWNH','2035-07-23','BE'),
('W7QCUV32S','2031-01-12','BE'),
('P7TZ55AMM','2033-04-06','B96'),
('V35K316CH','2028-06-17','B96'),
('93IWNT2AB','2027-10-22','BE'),
('4CXGFGZCI','2025-02-15','B'),
('IX1F058FT','2025-08-04','BE'),
('I8SK7XA1N','2033-02-07','BE'),
('YU32L58G7','2028-06-29','B'),
('3TJNL6Z8N','2034-07-13','B'),
('P61DJN07Q','2035-02-24','B96'),
('OYMS9P80R','2028-02-04','B96'),
('T5N45646K','2035-12-04','B96'),
('218PONE95','2027-03-01','B96'),
('UA1IERFB0','2035-09-28','BE'),
('0VXXDA51S','2027-11-11','B'),
('NVE8M4BSH','2031-09-07','B'),
('A2JASJPA0','2027-03-20','B96'),
('O2EERFGRH','2035-12-27','BE'),
('2WU44VDME','2033-12-13','B'),
('VX5WDTN1D','2026-10-22','B96'),
```

Turni

```
INSERT INTO Turni (OrarioInizio,OrarioFine) VALUES ('9','22'),
('11','20'),
('11','22'),
('10','21'),
('9','17');
```

Assicurazioni

```
INSERT INTO Assicurazioni (Numero,DDS,Tipo,Stato,Targa) VALUES ('0','2023-01-03','Furto','Scaduta','EK436DE'),
('1','2024-06-13','Incendio','Valida','EN727FF'),
('2','2024-04-17','Incendio','Valida','CK232BE'),
('3','2023-03-29','Polizza cristalli','Scaduta','FP955AD'),
('4','2024-08-01','Furto','Valida','CI117BB'),
('5','2024-07-15','Furto','Valida','FP987GA'),
('6','2023-06-27','Incendio','Scaduta','DM916FG'),
('7','2024-03-05','Kasko','Valida','CJ782BG'),
('8','2024-04-22','Incendio','Valida','GL141CB'),
('9','2023-04-03','Kasko','Scaduta','AL221FF'),
```

```
('10','2023-05-13','Base','Scaduta','BP938CE'),
('11','2024-04-05','Base','Valida','CQ586DD'),
('12','2023-09-17','Incendio','Scaduta','BL113GA'),
('13','2023-02-11','Furto','Scaduta','CQ555AG'),
('14','2023-03-20','Kasko','Scaduta','EK965AC'),
('15','2023-03-29','Incendio','Scaduta','CR139AB'),
('16','2023-06-26','Base','Scaduta','EH378CE'),
('17','2023-03-19','Polizza cristalli','Scaduta','EQ164EF'),
('18','2024-10-14','Incendio','Valida','FP821FB'),
('19','2024-05-14','Furto','Valida','DR927GG'),
('20','2024-10-14','Furto','Valida','BP716BA'),
('21','2024-06-08','Furto','Valida','BQ485ED'),
```

Veicoli

```
INSERT INTO Veicoli (Targa,Marca,Modello,NumPosti,Matricola) VALUES ('EK436DE','Alfa Romeo','Giulia','6','308673'),
('EN727FF','Audi','A1','5','334452'),
('CK232BE','Alfa Romeo','Giulia','8','792280'),
('FP955AD','Fiat','Punto','8','765293'),
('CI117BB','Renault','Clio','9','763099'),
('FP987GA','Audi','RS7','12','659075'),
('DM916FG','Audi','Q8','7','705297'),
('CJ782BG','Range Rover','Sport','4','211588'),
('GL141CB','Range Rover','Defender','3','418817'),
('AL221FF','Audi','Q3','8','031886'),
('BP938CE','Range Rover','Hybrid','12','781871'),
('CQ586DD','Fiat','Tipo','6','847719'),
('BL113GA','Fiat','Tipo','11','895729'),
('CQ555AG','Renault','Captur','10','680788'),
('EK965AC','Renault','Captur','5','835963'),
('CR139AB','Range Rover','Hybrid','12','329315'),
('EH378CE','Fiat','Panda','10','673447'),
('EQ164EF','BMW','X8','8','854550'),
('FP821FB','BMW','Gran Coupè','10','237246'),
('DR927GG','BMW','Gran Coupè','11','695896'),
('BP716BA','Alfa Romeo','Giulia','6','485164'),
('BQ485ED','Fiat','Punto','6','103105'),
('AQ062GA','BMW','X3','6','206053'),
('AI301GD','Alfa Romeo','Giulia','3','419980'),
('DR925ED','Range Rover','Sport','11','928720'),
('BP272AA','Range Rover','Hybrid','12','933775'),
('CQ221EF','Renault','Arkana','6','440415'),
```

Autisti

```
INSERT INTO Autisti (Matricola,Nome,Cognome,Email,DDN,NumeroTelefono,NumeroPatente,Stipendio)
VALUES ('803861','Ermanno','Treves','Ermanno.Treves@piacentini-foa.org','1980-01-13','+39
094203491','VZMX360RW','900'),
('674582','Arturo','Barbarigo','Arturo.Barbarigo@collina-castelli.com','1994-09-
05','322404041','X78P0G5NM','900'),
('054118','Gloria','Russo','Gloria.Russo@udinese.com','1981-07-21','+39
0963723013','TE9KHF73D','1100'),
('564135','Giustino','Mantegna','Giustino.Mantegna@murri-iannelli.it','1991-12-
20','0882895359','PMOKYCWNH','900'),
('674048','Gianfrancesco','Pasqua','Gianfrancesco.Pasqua@salata.it','1999-01-
02','3393948262','W7QCUV32S','900'),
('660172','Patrizio','Medici','Patrizio.Medici@aulenti.eu','1986-10-
26','048114730','P7TZ55AMM','1100'),
('523481','Fabia','Silvestri','Fabia.Silvestri@nitto.com','1987-09-
```

```

15','05440915708','V35K316CH','1100'),
('988530','Gloria','Benigni','Gloria.Benigni@cagnin.it','1986-06-
08','3534192949','93IWNT2AB','1200'),
('447279','Alderano','Cabrini','Alderano.Cabrini@orengo-nonis.com','1999-11-29','+39
041866389','4CXGFGZCI','800'),
('807679','Ermenegildo','Storladi','Ermenegildo.Storladi@lussu-rossetti.com','1996-09-22','+39
35198482539','IX1F058FT','1200'),
('431374','Isa','Priuli','Isa.Priuli@giulietti-guarana.it','1993-08-17','+39
018341037','I8SK7XA1N','800'),
('966073','Calcedonio','Speri','Calcedonio.Speri@onio.com','1980-04-13','+39
0372686902','YU32L58G7','900'),
('179627','Massimiliano','Gucci','Massimiliano.Gucci@pignatti-gatto.it','1993-08-24','+39
3626522141','3TJNL6Z8N','1100'),

```

Manutentori

```

INSERT INTO Manutentori (ID_Manutentore, Nome, Cognome, Email, DDN, NumeroTelefono, Qualifica) VALUES
('0','Arturo','Gradenigo','Arturo.Gradenigo@geraci.com','1984-04-
17','0535085250','Carrozziere'),
('1','Elladio','Borroni','Elladio.Borroni@goldstein.eu','1989-12-
10','3791055485','Elettrauto'),
('2','Gianluigi','Fogazzaro','Gianluigi.Fogazzaro@mancini.eu','1977-01-26','+39
057340992','Meccanico'),
('3','Susanna','Catenazzi','Susanna.Catenazzi@anichini.it','1995-01-19','+39
088144127','Carrozziere'),
('4','Tonino','Pizzamano','Tonino.Pizzamano@doria.eu','1990-12-15','+39
3285977712','Carrozziere'),
('5','Fabio','Andreozzi','Fabio.Andreozzi@angeli-chindamo.org','1990-12-08','+39
037282733','Gommista'),
('6','Iolanda','Scarpetta','Iolanda.Scarpetta@antonello.com','1978-08-
11','39743390554','Carrozziere'),
('7','Dolores','Ioppi','Dolores.Ioppi@morgagni-mattarella.net','1998-10-
17','3883691529','Carrozziere'),
('8','Pier','Ligorio','Pier.Ligorio@vercelloni.net','1997-03-26','0426305966','Elettrauto'),
('9','Achille','Andreotti','Achille.Andreotti@vivaldi.eu','1982-08-
16','324425451','Carrozziere'),
('10','Ettore','Galuppi','Ettore.Galuppi@argan.net','2000-12-10','09100100638','Meccanico'),
('11','Dante','Munari','Dante.Munari@muratori.org','1976-02-21','+39 0566126669','Elettrauto'),
('12','Flavio','Manolessa','Flavio.Manolessa@bandello.it','1990-08-
07','094122948','Meccanico'),
('13','Sabatino','Montalti','Sabatino.Montalti@miniati.it','1976-06-
14','3233577163','Elettrauto'),
('14','Gastone','Quasimodo','Gastone.Quasimodo@ossani.org','1987-09-28','+39
05875984919','Carrozziere'),
('15','Michele','Vespucci','Michele.Vespucci@querini.org','1997-06-01','+39
37794530162','Meccanico'),
('16','Raffaella','Venditti','Raffaella.Venditti@palladio.eu','1999-01-
24','0545473820','Elettrauto'),
('17','Gelsomina','Ricci','Gelsomina.Ricci@gozzi-carocci.it','1984-05-25','+39
0375563691','Meccanico'),

```

ContattaPerGuasto

```

INSERT INTO ContattaPerGuasto (ID_Manutentore, Matricola, Motivo, Data) VALUES
('77','826280','Errore centralina','2024-03-07'),
('29','280503','Guarnizione della testata bruciata','2024-04-20'),
('27','784173','Batteria scarica','2024-05-06'),
('137','085888','Errore centralina','2023-08-27'),
('79','461402','Spia dell motore accesa','2023-11-27'),
('16','534498','Cambio pasticche dei freni','2023-07-08'),

```

```
('109','349470','Differenziale rotto','2024-06-30'),  
('68','326310','Errore centralina','2023-07-30'),  
('122','606756','Differenziale rotto','2024-11-27'),  
('68','913781','Radiatore bucato','2023-03-17'),  
('108','826280','Specchietto rotto','2023-10-15'),  
('181','523481','Spia dell motore accesa','2024-11-25'),  
('50','895222','Radiatore bucato','2024-04-10'),
```

Utenti

```
INSERT INTO Utenti (ID_Utente, Nome, Cognome, Email, Password, DDN) VALUES  
('0','Emilio','Oliboni','Emilio.Oliboni@pascarella-sordi.com','FbAt9bmU6','1989-06-01'),  
('1','Beatrice','Mazzini','Beatrice.Mazzini@gadda.it','mybw1hMRh','1992-01-31'),  
('2','Lando','Cadorna','Lando.Cadorna@gregorio.net','S4IGo5i03','1976-08-30'),  
('3','Torquato','Callegaro','Torquato.Callegaro@capone.it','v5iMDlWUa','1995-09-14'),  
('4','Barbara','Sgalambro','Barbara.Sgalambro@cusano.com','oGbRz2V1X','1998-06-15'),  
('5','Gianfrancesco','Persico','Gianfrancesco.Persico@togliatti.eu','cIdzMUMkT','1996-01-21'),  
('6','Venancio','Gentilini','Venancio.Gentilini@ponti.it','dNDTjSwZ0','1982-06-24'),  
('7','Patrizio','Altera','Patrizio.Altera@gritti-bellucci.it','NFZoaQd2W','1994-11-19'),  
('8','Ippazio','Passalacqua','Ippazio.Passalacqua@giannelli.eu','Ul8a4GoyN','1987-12-11'),  
('9','Giuseppina','Valentino','Giuseppina.Valentino@pace.eu','Ni03oqY48','1992-05-21'),  
('10','Baccio','Casarin','Baccio.Casarin@modigliani-sabatini.com','3ZpNuKkZz','1998-06-08'),  
('11','Cassandra','Dossetti','Cassandra.Dossetti@gianinazzi.it','EhaC0cLNa','1997-03-11'),  
('12','Germana','Parpinel','Germana.Parpinel@stefanelli.it','VtUJl4PeK','1990-10-09'),  
('13','Ettore','Paolucci','Ettore.Paolucci@carocci.com','k8cLK5LSM','1986-04-18'),  
('14','Lilla','Fracci','Lilla.Fracci@barillaro.net','q4akiSdgB','1988-08-03'),  
('15','Cristina','Ferrazzi','Cristina.Ferrazzi@vigorelli-boito.eu','pruKldMAL','1981-10-14'),  
('16','Adriana','Antonucci','Adriana.Antonucci@borsellino.it','mFmQjLUUx','1999-01-13'),  
('17','Paola','Brunelleschi','Paola.Brunelleschi@proietti-tasso.it','Yzex8NyEa','1988-12-27'),  
('18','Eugenia','Papafava','Eugenia.Papafava@tiepolo-bettin.com','OMRZBS7yF','1982-10-22'),  
('19','Piero','Piacentini','Piero.Piacentini@granatelli.org','BSKB5tk1x','1998-08-03'),  
('20','Riccardo','Venditti','Riccardo.Venditti@curci.eu','LLdE30YxS','1985-09-23'),  
('21','Giancarlo','Savorgnan','Giancarlo.Savorgnan@jilani-bonaventura.it','2BEfSzt3k','1986-08-10'),
```

Carta

```
INSERT INTO Carta (NumeroCarta, DataScadenza, CVV, ID_Utente) VALUES ('5419 3484 2461 3362','2031-10-09','684','6961'),  
('4029 9017 0495 8697','2028-05-18','664','6924'),  
('4607 6953 9239 2530','2027-08-22','885','8673'),  
('4558 5489 4230 1191','2032-12-19','883','7777'),  
('5492 4233 2942 3887','2030-12-02','913','2454'),  
('5477 6017 0799 6651','2034-11-29','244','4348'),  
('5379 4363 5968 2720','2033-10-13','942','2764'),  
('4450 1658 0372 7289','2030-02-02','806','2759'),  
('4960 4473 7840 6693','2032-02-16','221','8264'),  
('4331 7076 2209 9857','2032-08-18','542','4720'),  
('5284 4385 5153 8849','2030-08-01','189','7278'),  
('4983 2619 2205 1289','2028-01-01','007','5676'),  
('5899 0654 5175 1796','2030-08-16','613','3121'),  
('4112 5895 5886 6938','2030-07-14','335','9718'),  
('4833 2182 4762 9538','2027-12-26','909','4547'),  
('5436 5174 3158 8021','2028-07-25','885','8365'),  
('4207 9361 4153 4064','2034-07-26','151','1933'),  
('5044 5623 9877 9468','2033-08-22','057','4011'),  
('5282 0605 2972 3151','2027-07-11','436','1557'),  
('5077 5943 4127 7775','2029-08-23','893','7256'),
```

Fermate

```

INSERT INTO Fermate (NomeFermata,Latitudine,Longitudine) VALUES
('Anagnina','41.8425652','12.5860085'),
('Termini','41.9016577','12.5007858'),
('Giardinetti','41.8641331','12.6105711'),
('Lucio Sestio','41.8596969','12.5572829'),
('Porta Furba','41.863995','12.5443672'),
('Tor Bella Monaca','41.868496','12.6412461'),
('Campo de Fiori','41.8955774','12.472158637200002'),
('Trastevere','41.8911586','12.466845904466918'),
('Tufello','42.5621714','1.3071746'),
('Pigneto','41.8884916','12.5281664'),
('Palmiro Togliatti','41.9033311','12.5741939'),
('Salaria','42.912642649999995','13.885354808053245'),
('Verano','46.6053657','11.2262711'),
('Prima Porta','42.0019746','12.4859701'),
('Colosseo','41.8902614','12.493087103595503'),
('Prenestina','41.8953602','12.6149407');

```

Richieste Prenotazioni

```

INSERT INTO RichiestePrenotazioni
(ID_Utente,Partenza,Arrivo,DataRichiesta,OrarioRichiesta,NumeroPasseggeri) VALUES ('5831','Tor
Bella Monaca','Colosseo','2022-01-10','14','1'),
('3323','Prima Porta','Verano','2024-02-01','10','11'),
('2354','Prenestina','Lucio Sestio','2022-01-11','21','2'),
('7801','Trastevere','Campo de Fiori','2021-01-14','22','2'),
('5322','Palmiro Togliatti','Lucio Sestio','2022-05-30','15','10'),
('8642','Lucio Sestio','Verano','2023-05-10','21','4'),
('202','Tor Bella Monaca','Prima Porta','2020-11-03','10','4'),
('4230','Tufello','Lucio Sestio','2023-12-24','20','8'),
('2416','Verano','Palmiro Togliatti','2022-05-20','10','1'),
('5839','Trastevere','Pigneto','2022-02-06','9','9'),
('7600','Campo de Fiori','Anagnina','2022-07-01','22','12'),
('9537','Anagnina','Termini','2023-09-14','10','6'),
('6966','Colosseo','Campo de Fiori','2021-08-04','16','9'),
('7507','Prima Porta','Salaria','2021-04-10','11','10'),
('564','Trastevere','Pigneto','2021-07-27','9','8'),
('1074','Salaria','Colosseo','2021-11-04','22','7'),
('198','Termini','Pigneto','2020-03-07','20','12'),
('4501','Tufello','Porta Furba','2020-03-19','20','4'),

```

Tratte Completate

```

INSERT INTO TratteCompletate
(ID_Utente,Partenza,Arrivo,DataRichiesta,OrarioRichiesta,Costo,MetodoDiPagamento,DataPagamento,
OraPagamento,Autista) VALUES ('5831','Tor Bella Monaca','Colosseo','2022-01-
10','14','65','Carta di debito','2022-01-10','23','930071'),
('3323','Prima Porta','Verano','2024-02-01','10','65','Carta di credito','2024-02-
01','22','392512'),
('2354','Prenestina','Lucio Sestio','2022-01-11','21','65','Postepay','2022-01-
11','23','579400'),
('7801','Trastevere','Campo de Fiori','2021-01-14','22','50','CashUp','2021-01-
14','23','538604'),
('5322','Palmiro Togliatti','Lucio Sestio','2022-05-30','15','65','Contanti','2022-05-
30','23','176323'),
('8642','Lucio Sestio','Verano','2023-05-10','21','25','Carta di debito','2023-05-
10','23','912945'),
('202','Tor Bella Monaca','Prima Porta','2020-11-03','10','50','Paypal','2020-11-
03','11','910588'),
('4230','Tufello','Lucio Sestio','2023-12-24','20','65','Carta di credito','2023-12-

```

```

24','22','606756'),
('2416','Verano','Palmiro Togliatti','2022-05-20','10','115','Contanti','2022-05-
20','14','847781'),
('5839','Trastevere','Pigneto','2022-02-06','9','50','Carta di debito','2022-02-
06','23','424170'),

```

Feedback

```

INSERT INTO Feedback
(ID_Feedback,StelleUtente,CommentoUtente,StelleAutista,CommentoAutista,ID_Utente,Partenza,Arriv
o,DataRichiesta,OrarioRichiesta) VALUES ('0','5','Autista veramente cordiale','2','Stava
fumando in macchina','141','Prima Porta','Verano','2023-06-11','15'),
('1','2','Non mi è piaciuto lo stile di guida','1','Utente
scortese!','3246','Colosseo','Termini','2022-03-13','14'),
('2','5','Autista veramente cordiale','4','Utente gentile','3501','Palmiro
Togliatti','Trastevere','2021-08-31','10'),
('3','4','Veicolo molto pulito e comodo.','2','Non rispetta l
autista','8114','Pigneto','Salaria','2023-09-19','20'),
('4','3','Tutto nella norma','4','Utente rispettoso.','4479','Giardinetti','Tor Bella
Monaca','2021-07-07','21'),
('5','4','Esperienza normale','2','Utente ritardatario','3928','Trastevere','Porta
Furba','2021-10-20','20'),
('6','5','Ottima esperienza, lo dirò a tutti','3','Utente ok','2069','Termini','Campo de
Fiori','2022-09-02','22'),
('7','4','Veicolo molto pulito e comodo.','3','Utente ok','9438','Colosseo','Termini','2023-05-
02','14'),
('8','1','Guidava in stato di ebrezza','1','L utente insisteva nel cambiare
strada','646','Palmiro Togliatti','Pigneto','2022-09-16','14'),
('9','3','Nulla di particolare','4','Utente gentile','270','Pigneto','Porta Furba','2023-02-
15','11'),
('10','5','Ottima esperienza, lo dirò a tutti','5','Molto bravo e
cortese','5736','Verano','Campo de Fiori','2021-02-01','15'),
('11','5','Autista veramente cordiale','2','Stava fumando in macchina','5441','Lucio
Sestio','Prima Porta','2022-01-13','22'),
('12','1','Esperienza orribile','1','L utente offende','6535','Porta Furba','Anagnina','2023-
10-23','9'),
('13','5','Autista veramente cordiale','2','Non rispetta l
autista','5231','Pigneto','Colosseo','2022-12-20','9'),

```

Tratte Rifiutate

```

INSERT INTO TratteRifiutate
(ID_Utente,Partenza,Arrivo,DataRichiesta,OrarioRichiesta,Motivazione,Autista) VALUES
('5241','Campo de Fiori','Prenestina','2023-05-21','20','Problema generale','335757'),
('1049','Giardinetti','Campo de Fiori','2022-11-27','20','Utente con recensioni troppo
negative','436909'),
('2965','Termini','Lucio Sestio','2020-11-15','9','Indisponibilità al servizio','617360'),
('7040','Termini','Tor Bella Monaca','2022-06-12','20','Utente con recensioni troppo
negative','976651'),
('5893','Palmiro Togliatti','Prenestina','2023-11-21','11','Indisponibilità al
servizio','246885'),
('1276','Prenestina','Tor Bella Monaca','2022-03-16','10','Utente con recensioni troppo
negative','206858'),
('8823','Termini','Trastevere','2021-07-24','14','Fuori dal mio orario lavorativo','951021'),
('2202','Giardinetti','Verano','2022-06-10','20','Indisponibilità al servizio','234528'),
('2979','Tor Bella Monaca','Pigneto','2020-06-12','15','Fuori dal mio orario
lavorativo','553982'),
('8373','Porta Furba','Prima Porta','2021-07-14','21','Fuori dal mio orario
lavorativo','992435'),
('7380','Anagnina','Pigneto','2021-09-11','21','Indisponibilità al servizio','072980'),

```

```

('3738','Campo de Fiori','Termini','2022-09-13','15','Problema generale','426252'),
('2415','Pigneto','Lucio Sestio','2020-09-29','22','Fuori dal mio orario lavorativo','914178'),
('8082','Trastevere','Tor Bella Monaca','2022-08-06','15','Troppo lontano','878298'),
('3923','Tufello','Porta Furba','2020-02-21','20','Utente con recensioni troppo negative','956283'),
('4097','Lucio Sestio','Pigneto','2020-05-09','22','Problema generale','932216'),
('4688','Termini','Pigneto','2022-04-18','15','Utente con recensioni troppo negative','341191'),
('5074','Tor Bella Monaca','Palmo Togliatti','2021-07-24','11','Fuori dal mio orario lavorativo','133545'),
('8408','Termini','Anagnina','2022-09-12','21','Problema generale','861217'),
('5892','Giardinetti','Porta Furba','2023-11-30','22','Problema generale','888331'),

```

Script di creazione automatica di query

Per rendere paragonabili i tempi di esecuzione delle query non ottimizzate con quelle ottimizzate, è stato necessario introdurre, nel database, un gran numero di record.

Per velocizzare questo processo, è stato scritto uno script in Python che scrive tutti gli inserimenti generati casualmente, in un semplice file di testo.

Il file è il seguente

```

import random
from faker import Faker
import string
import decimal
import datetime
from geopy.geocoders import Nominatim

def getLatAndLong(posto):
    # calling the Nominatim tool and create Nominatim class
    loc = Nominatim(user_agent="Geopy Library")

    # entering the location name
    getLoc = loc.geocode(posto)

    return getLoc.latitude, getLoc.longitude

def prendi_due_elementi(array):
    # Scegli due indici casuali
    indice1, indice2 = random.sample(range(len(array)), 2)

    # Se gli elementi sono uguali, scegli un nuovo indice2
    while array[indice1] == array[indice2]:
        indice2 = random.randint(0, len(array) - 1)

    return array[indice1], array[indice2]

fake = Faker("it_IT")

#Funzione Rand.DDN
def genRandomDate():
    start_date = datetime.date(1975, 1, 1)
    end_date = datetime.date(2001, 12, 30)
    num_days = (end_date - start_date).days
    rand_days = random.randint(1, num_days)
    random_date = start_date + datetime.timedelta(days=rand_days)

    return random_date

```

```

#Funzione Rand.DA
def genRandomInsuranceDate():
    start_date = datetime.date(2023, 1, 1)
    end_date = datetime.date(2025, 1, 1)
    num_days = (end_date - start_date).days
    rand_days = random.randint(1, num_days)
    random_date = start_date + datetime.timedelta(days=rand_days)

    return random_date

#Funzione Rand.CD
def genRandomCardDate():
    start_date = datetime.date(2027, 1, 1)
    end_date = datetime.date(2034, 12, 30)
    num_days = (end_date - start_date).days
    rand_days = random.randint(1, num_days)
    random_date = start_date + datetime.timedelta(days=rand_days)

    return random_date

#Funzione Rand.PD
def genRandomLicenceDate():
    start_date = datetime.date(2025, 1, 1)
    end_date = datetime.date(2035, 12, 30)
    num_days = (end_date - start_date).days
    rand_days = random.randint(1, num_days)
    random_date = start_date + datetime.timedelta(days=rand_days)

    return random_date

#Funzione Rand.DDR
def genRandomRequestDate():
    start_date = datetime.date(2022, 1, 1)
    end_date = datetime.date(2023, 12, 30)
    num_days = (end_date - start_date).days
    rand_days = random.randint(1, num_days)
    random_date = start_date + datetime.timedelta(days=rand_days)

    return random_date

#Funzione Rand.Mail
def generateEmail(name, surname):

    domain = fake.domain_name()

    return f"{name}.{surname}@{domain}"

#Funzione Rand.T
def generateTarga():

    SYMBOLS = "ABCDEFG"
    SYMBOLS_END = "HIJKLMNOPQR"
    NUMBERS = "0123456789"
    start = "".join(random.choice(SYMBOLS) for i in range(1))
    start_2 = "".join(random.choice(SYMBOLS_END) for i in range(1))
    mezzo = "".join(random.choice(NUMBERS) for i in range(3))
    fine = "".join(random.choice(SYMBOLS) for i in range(2))

    return start+start_2+mezzo+fine

#Funzione Rand.PSW
def generatePsw():

    ALL = "0123456789ABCDEFGHIJKLMNPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz"

```

```

psw = "".join(random.choice(ALL) for i in range(9))
return psw
#Funzione Rand.CN
def generateCardNumber():
    NUMBERS = "0123456789"
    number = "".join(random.choice(NUMBERS) for i in range(16))
    return number
#Funzione Rand.Star
def checkStelleUtenti(stelle):
    if stelle == 1:
        return 1
    elif stelle == 2:
        return 2
    elif stelle == 3:
        return 3
    elif stelle == 4:
        return 4
    elif stelle == 5:
        return 5

print("Inizio esecuzione...")

print("L'ORDINE DI ESECUZIONE DEI FILE È 1.txt,2.txt,etc...")

print("Inizio Creazione 1.txt")

SYMBOLS = "ABCDEFGHIJKLMNPQRSTUVWXYZ"
NUMBERS = "0123456789"
ALL_SYMBOLS = "0123456789ABCDEFGHIJKLMNPQRSTUVWXYZ"
print("Inizio Creazione 2.txt")

f = open("2.txt","w+")

print("----- Inizio Inserimento Patente\n")

patenti = ["B","BE","B96"]
unique_Patente = []
values_patenti = []
for i in range(3000):
    data = genRandomLicenceDate()
    random_numpatente = "".join(random.choice(ALL_SYMBOLS) for i in range(9))
    categoria = "".join(random.choice(patenti) for i in range(1))
    unique_Patente.append(random_numpatente)

    query = "(" + random_numpatente + ", " + str(data) + ", " + categoria + ")"

    values_patenti.append(query)
f.write(
    "INSERT INTO Patente (NumeroPatente,DDS,Categoria) VALUES "+",".join(values_patenti)+";"
)
f.write("\n")
print("----- Fine Inserimento Patente\n")
f.write("\n")
print("----- Inizio Inserimento Turni\n")

unique_Turno = []
ora_inizio = ['9','10','11','14']
ora_fine = ['17','20','21','22']
values_turni = []
for i in range(5):

    #random_turno = "".join(random.choice(NUMBERS) for i in range(1))

```

```

inizio = "".join(random.choice(ora_inizio))
fine = "".join(random.choice(ora_fine))
unique_Turno.append((inizio,fine))
query = "(""+ inizio+ '", "'+ fine+ "')"

values_turni.append(query)
f.write(
    "INSERT INTO Turni (OrarioInizio,OrarioFine) VALUES "+",\n".join(values_turni)+";"
)
f.write("\n")
print("----- Fine Inserimento Turni\n")
print("----- Inizio Inserimento Autisti\n")

unique_Autisti = []
values_autisti = []
stipendio = ["1200", "1100", "900", "800"]

for i in range(3000):
    matricola = "".join(random.choice(NUMBERS) for i in range(6))
    if matricola in unique_Autisti:
        matricola = "".join(random.choice(NUMBERS) for i in range(6))
    random_patente = unique_Patente[i]
    #random_Turno = random.choice(unique_Turno[1:])
    #random_targa = random.choice(unique_Veicolo)
    nome = fake.first_name()
    cognome = fake.last_name()
    email = generateEmail(nome,cognome)
    ddn = genRandomDate()
    num_telefono = fake.phone_number()
    query = "(""+ matricola+ '", "'+ str(nome)+ '", "'+ str(cognome)+ '", "'+ str(email)+ '", "'+
str(ddn)+ '", "'+ num_telefono+ '", "'+ random_patente+ '", "'+random.choice(stipendio)+")"
    unique_Autisti.append(matricola)
    values_autisti.append(query)
f.write(
    "INSERT INTO Autisti
(Matricola,Nome,Cognome,Email,DDN,NumeroTelefono,NumeroPatente,Stipendio) VALUES
"+",\n".join(values_autisti)+";"
)
f.write("\n")
print("----- Fine Inserimento Autisti\n")
f.write("\n")
print("----- Inizio Inserimento Veicoli\n")
unique_Veicolo = []
values_veicolo = []
dict_veicoli={

    "Fiat": ["Punto", "Panda"],
    "BMW": ["Q3", "Q8", "X1", "Gran Coupè"],
    "Audi": ["RS7"],
    "Range Rover": ["Hybrid", "Defender", "Sport"]
}

for i in range(3000):
    random_targa = generateTarga()
    #random_assicurazione = unique_Assicurazione[i]
    marca_random = random.choice(list(dict_veicoli.keys()))
    modello_random = str(random.choice(dict_veicoli[marca_random]))
    autista = unique_Autisti[i]
    query = "("+ str(random_targa)+ '", "'+ str(marca_random)+ '", "'+ str(modello_random)+
'", "'+str(random.randint(3,12))+", "'+autista+")"
    unique_Veicolo.append(random_targa)
    values_veicolo.append(query)

```

```

f.write(
    "INSERT INTO Veicoli (Targa,Marca,Modello,NumPosti,Matricola) VALUES
"+",\n".join(values_veicolo)+";"
)
f.write("\n")
print("----- Fine Inserimento Veicoli\n")
print("----- Inizio Inserimento Assicurazione\n")

unique_Assicurazione = []
values_assicurazione = []
tipo_assicurazione=[ "Kasko", "Furto", "Incendio", "Base", "Polizza cristalli"]

for i in range(3000):
    random_id = str(i)
    targa = unique_Veicolo[i]
    data = genRandomInsuranceDate()
    tipo = random.choice(tipo_assicurazione)
    if str(data) < "2024-02-16":
        stato = "Scaduta"
    else:
        stato = "Valida"
    query = "(" + str(random_id) + ", '" + str(data) + "', '" + str(tipo) + "', '" +
str(stato) + "', '" + str(targa) + "')"
    unique_Assicurazione.append(random_id)
    values_assicurazione.append(query)
f.write(
    "INSERT INTO Assicurazioni (Numero,DDS,Tipo,Stato,Targa) VALUES
"+",\n".join(values_assicurazione)+";"
)
f.write("\n")
print("----- Fine Inserimento Assicurazione\n")
f.write("\n")
f.write("\n")
f.write("\n")
print("----- Inizio Inserimento TabellaOrarioLavorativo\n")

values_tabella = []
unique_TabellaOrario = []
for i in range(2000):
    matricola = random.choice(unique_Autisti)
    turno = random.choice(unique_Turno)
    turno_inizio = turno[0]
    turno_fine = turno[1]
    data = genRandomInsuranceDate()
    query = "(" + matricola + ", '" + turno_inizio + "', '" + turno_fine + "', '" + str(data) + "')"
    unique_TabellaOrario.append((matricola,turno_inizio,turno_fine,data))
    values_tabella.append(query)
f.write(
    "INSERT INTO TabellaOrarioLavorativo (Matricola,OraInizio,OraFine,Data) VALUES
"+",\n".join(values_tabella)+";"
)
f.write("\n")
print("----- Fine Inserimento TabellaOrarioLavorativo\n")
f.write("\n")
print("----- Inizio Inserimento Manutentori\n")

unique_Manutentori = []
values_manutentori = []
qualifica = [ "Gommista", "Elettrauto", "Meccanico", "Carrozziere"]
for i in range(200):
    random_id = str(i)
    nome = fake.first_name()

```

```

cognome = fake.last_name()
email = generateEmail(nome,cognome)
ddn = genRandomDate()
query = "(" + random_id + ", " + random.choice(qualifica) + ")"
telefono = fake.phone_number()
query = "(" + random_id + ", " + str(nome) + ", " + str(cognome) + ", " + str(email) + ", " +
str(ddn) + ", " + str(telefono) + ", " + random.choice(qualifica) + ")"
unique_Manutentori.append(random_id)
values_manutentori.append(query)
f.write(
    "INSERT INTO Manutentori (ID_Manutentore, Nome, Cognome, Email, DDN, NumeroTelefono, Qualifica)
VALUES " + "\n".join(values_manutentori) + ";" )
)
f.write("\n")
print("----- Fine Inserimento Manutentori\n")
f.write("\n")
print("----- Inizio Inserimento ContattaPerGuasto\n")

unique_Contatto = []
values_contatto = []
motivi = ["Gomma Bucata", "Spia dell motore accesa", "Radiatore bucato", "Batteria scarica", "Problema con il FAP", "Errore centralina", "Specchietto rotto", "Guarnizione della testata bruciata", "Rottura degli ammortizzatori", "Semiasse distrutto", "Differenziale rotto", "La macchina non parte", "Cambio pasticche dei freni"]

for i in range(1500):
    random_manutentore = random.choice(unique_Manutentori)
    random_autista = random.choice(unique_Autisti[0:200])
    data = genRandomInsuranceDate()
    query = "(" + random_manutentore + ", " + random_autista +
", " + random.choice(motivi) + ", " + str(data) + ")"
    unique_Contatto.append((random_manutentore, random_autista))
    values_contatto.append(query)
f.write(
    "INSERT INTO ContattaPerGuasto (ID_Manutentore, Matricola, Motivo, Data) VALUES
" + "\n".join(values_contatto) + ";" )
)
f.write("\n")
print("----- Fine Inserimento ContattaPerGuasto\n")

print("2.txt Done")
f.close()

print("Inizio Creazione 3.txt")
f = open("3.txt", "w+")

f.write("\n")
print("----- Inizio Inserimento Utenti\n")

unique_Utenti = []
values_utenti = []

for i in range(10000):
    random_id = str(i)
    surname = fake.last_name()
    name = fake.first_name()
    email = str(generateEmail(name, surname))
    psw = generatePsw()
    data = genRandomDate()
    query = "(" + random_id + ", " + name + ", " + surname +
", " + email + ", " + psw + ", " + str(data) + ")"
    unique_Utenti.append(random_id)

```

```

values_utenti.append(query)
f.write(
    "INSERT INTO Utenti (ID_Utente,Nome,Cognome,Email,Password,DDN) VALUES
"+",\n".join(values_utenti)+";"
)
f.write("\n")
print("----- Fine Inserimento Utenti\n")
f.write("\n")
print("----- Inizio Inserimento Carte\n")

unique_Carta = []
values_carta = []

utente_carta = []
for i in range(20000):

    numero_Carta = str(random.randint(4,5))+"".join(str(random.randint(0,9)) for i in
range(3))+""+"".join(str(random.randint(0,9)) for i in range(4))+"
"+"".join(str(random.randint(0,9)) for i in range(4))+""+"".join(str(random.randint(0,9)) for
i in range(4))
    data_scadenza = genRandomCardDate()
    cvv = "".join(str(random.randint(0,9)) for i in range(3))
    utente = random.choice(unique_Utenti)
    query = "("+ numero_Carta+ ", "+ str(data_scadenza)+ ", "+ cvv+ ", "+utente+ ")"
    unique_Carta.append(numero_Carta)
    values_carta.append(query)

    utente_carta.append((utente,numero_Carta))
f.write(
    "INSERT INTO Carta (NumeroCarta,DataScadenza,CVV,ID_Utente) VALUES
"+",\n".join(values_carta)+";"
)
print("----- Fine Inserimento Carte\n")

print("3.txt Done")
f.close()
print("Inizio creazione 4.txt")
f = open("4.txt","w+")
f.write("\n")
print("----- Inizio Inserimento Fermate\n")

unique_Fermata = []
values_fermata = []
fermata = ["Anagnina","Termini","Giardinetti","Lucio Sestio","Porta Furba","Tor Bella
Monaca","Campo de Fiori","Trastevere","Tufello","Pigneto","Palmoiro
Togliatti","Salaria","Verano","Prima Porta","Colosseo","Prenestina"]

for i in range(16):

    fermata_choice = fermata[i]
    latitudine,longitudine = getLatAndLong(fermata_choice)
    query = "("+ fermata_choice+ ", "+ str(latitudine)+ ", "+ str(longitudine)+ ")"
    unique_Fermata.append(fermata_choice)
    values_fermata.append(query)

f.write(
    "INSERT INTO Fermate (NomeFermata,Latitudine,Longitudine) VALUES
"+",\n".join(values_fermata)+";"
)
f.write("\n")
print("----- Fine Inserimento Fermate\n")

```

```

f.write("\n")
print("----- Inizio Inserimento RichiestaPrenotazioni\n")

unique_RichPren = []
values_ricpren = []
date = []
ora = ['9','10','11','14','15','16','20','21','22']
id_carta_utente = []
for i in range(20000):
    #random_id = str(i)
    passeggeri = str(random.randint(1,12))

    utente = random.choice(unique_Utenti)
    #autista = random.choice(unique_Autisti)
    data = genRandomRequestDate()
    orario = random.choice(ora)
    partenza,arrivo = prendi_due_elementi(unique_Fermata)

    query = "("+str(utente)+",'"+ str(partenza)+ "','"+ str(arrivo)+\
"',','"+ str(data)+"','"+ str(orario)+"','"+ str(passeggeri)+"')"
    unique_RichPren.append((utente,partenza,arrivo,data,orario))
    values_ricpren.append(query)
    date.append(data)
f.write(
    "INSERT INTO RichiestePrenotazioni
(ID_Utente,Partenza,Arrivo,DataRichiesta,OrarioRichiesta,NumerоПассажери) VALUES
#+,\n".join(values_ricpren)+";"
)
f.write("\n")
print("----- Fine Inserimento RichiestaPrenotazioni\n")
f.write("\n")
print("----- Inizio Inserimento TratteCompletate\n")

unique_TrattaC = []
values_trattac = []
costo = ["25","65","115","35","50"]
pagamento = ["Carta di credito","Paypal","Contanti","Satispay","Carta di
debito","CashUp","Postepay"]
for i in range(15000):
    pk = unique_RichPren[i]
    costi = random.choice(costo)
    orario_pagamento = random.choice(ora)
    if orario_pagamento < pk[4]:
        orario_pagamento = "23"
    metodo = random.choice(pagamento)
    autista = random.choice(unique_Autisti)
    query = "("+ str(pk[0])+ "','" + str(pk[1])+ "','" + str(pk[2])+ "','" + str(pk[3])+ "','" + str(pk[4])+\
"',','"+ str(costi)+"','"+ str(metodo)+"','"+ str(pk[3])+"','"+ str(orario_pagamento)+"','"+ str(autista)+"')"

    unique_TrattaC.append((utente,partenza,arrivo,data,orario))
    values_trattac.append(query)

f.write(
    "INSERT INTO TratteCompletate
(ID_Utente,Partenza,Arrivo,DataRichiesta,OrarioRichiesta,Costo,MetodoDiPagamento,DataPagamento,
OraPagamento,Autista) VALUES "+,\n".join(values_trattac)+";"
)
f.write("\n")
print("----- Fine Inserimento TratteCompletate\n")
f.write("\n")
print("----- Inizio Inserimento Feedback\n")

```

```

unique_Feed = []
values_feed = []
feedback_utente = {
    1: ["Non lo prenderò mai più!","Esperienza orribile","Guidava in stato di ebrezza"],
    2: ["Non mi è piaciuto lo stile di guida","La prossima volta preferirei un\' altro autista","Non guidava in modo sicuro"],
    3: ["Nulla di particolare","Tutto nella norma"],
    4: ["Veicolo molto pulito e comodo.","Esperienza normale"],
    5: ["Autista veramente cordiale","Ottima esperienza, lo dirò a tutti"],
}
feedback_autisti = {
    1: ["Utente scortese!","L\' utente offende","L\' utente insisteva nel cambiare strada"],
    2: ["Utente ritardatario","Non rispetta l'autista","Stava fumando in macchina"],
    3: ["Nulla di particolare","Utente ok"],
    4: ["Utente rispettoso.","Utente gentile"],
    5: ["Utente veramente genuino","Molto bravo e cortese"],
}

for i in range(15000):
    random_id = str(i)

    stelle_random_ut = random.choice(list(feedback_utente.keys()))
    commento_ut = str(random.choice(feedback_utente[stelle_random_ut]))

    #stelle_random_aut = checkStelleUtenti(stelle_random_ut)
    stelle_random_aut = random.choice(list(feedback_autisti.keys()))

    commento_aut = str(random.choice(feedback_autisti[stelle_random_aut]))
    fk_trattac = random.choice(unique_TrattaC)
    query = "(" + random_id + ", " + str(stelle_random_ut) + ", " + str(commento_ut) +
", " + str(stelle_random_aut) + ", " + str(commento_aut) + ", " + str(fk_trattac[0]) + ", " + str(fk_trattac[1]) + ", " + str(fk_trattac[2]) + ", " + str(fk_trattac[3]) + ", " + str(fk_trattac[4]) + ")"

    unique_Feed.append(random_id)
    values_feed.append(query)
    unique_TrattaC.remove(fk_trattac)

f.write(
    "INSERT INTO Feedback
(ID_Feedback,StelleUtente,CommentoUtente,StelleAutista,CommentoAutista,ID_Utente,Partenza,Arrivo,DataRichiesta,OrarioRichiesta) VALUES "+",\n".join(values_feed)+";"
)
f.write("\n")
print("----- Fine Inserimento Feedback\n")
f.write("\n")
print("----- Inizio Inserimento TratteRifiutate\n")

unique_TrattaR = []
values_trattar = []
motivi = ["Problema generale","Indisponibilità al servizio","Troppo lontano","Fuori dal mio orario lavorativo","Utente con recensioni troppo negative"]
for i in range(5000):
    fk_prenotazione = unique_RichPren[15000+i]
    motivo = random.choice(motivi)
    autista = random.choice(unique_Autisti)
    query = "(" + str(fk_prenotazione[0]) + ", " + str(fk_prenotazione[1]) + ", " +
"
```

```

str(fk_prenotazione[2])+ "!',!"+ str(fk_prenotazione[3])+ "!',!"+ str(fk_prenotazione[4])+ "!',!"+
str(motivo)+ "!',!"+str(autista)+"!)"
unique_TrattaR.append(fk_prenotazione)
values_trattar.append(query)
f.write(
    "INSERT INTO TratteRifiutate
(ID_Utente,Partenza,Arrivo,DataRichiesta,OrarioRichiesta,Motivazione,Autista) VALUES
"+,\n".join(values_trattar)+";"
)
f.write("\n")
print("----- Fine Inserimento TratteRifiutate\n")
print("4.txt Done")
f.close()

```

Query

- Visualizza tutte le tratte completate, con annesso costo e carta usata per il pagamento, fatte da un determinato utente

```

SELECT tc.*
from TratteCompletate tc
JOIN Utenti u on tc.ID_Utente = u.ID_Utente
WHERE u.Nome = 'Patrizio' AND u.Cognome = 'Altera';

```

ID_Utente	Partenza	Arrivo	DataRichiesta	OrarioRichiesta	Costo	MetodoDiPagamento	DataPagamento	OraPagamento	Autista
7	Palmiro Togliatti	Prima Porta	2021-05-17	16	65	Postepay	2021-05-17	20	643456
5789	Colosseo	Campo dei Fiori	2023-10-22	9	35	CashUp	2023-10-22	23	321560

- Visualizza tutti i veicoli la cui assicurazione scadrà entro febbraio 2024

```

SELECT a.Targa, a.DDS AS DataScadenza, a.Stato, a2.Nome,a2.Cognome,a2.Email
FROM Assicurazioni a
JOIN Veicoli v ON v.Targa = a.Targa
JOIN Autisti a2 on v.Matricola = a2.Matricola
WHERE YEAR(a.DDS) = "2024" AND MONTH(a.DDS) = "02";

```

Targa	DataScadenza	Stato	Nome	Cognome	Email
FP804AB	2024-02-04	Scaduta	Virginia	Binaghi	Virginia.Binaghi@andreeozzi.com
BQ424BB	2024-02-06	Scaduta	Rita	Brenna	Rita.Brenna@ligorio.eu
DH605EE	2024-02-17	Valida	Luciano	Civaschi	Luciano.Civaschi@toscani.eu
FI565CE	2024-02-04	Scaduta	Nina	Rusticucci	Nina.Rusticucci@peranda.eu
DK465DA	2024-02-10	Scaduta	Ignazio	Piane	Ignazio.Piane@scaramucci.com
CK943BC	2024-02-28	Valida	Federigo	Milanesi	Federigo.Milanesi@zecchini-golgi.it
CJ961EC	2024-02-25	Valida	Licia	Beccheria	Licia.Beccheria@germano.com
AQ386GD	2024-02-24	Valida	Fabia	Sonnino	Fabia.Sonnino@pacomio-cadorna.it
EJ494DG	2024-02-17	Valida	Fabia	Sonnino	Fabia.Sonnino@pacomio-cadorna.it
FH127CE	2024-02-25	Valida	Nedda	Babati	Nedda.Babati@camuccini.com
DJ749FF	2024-02-17	Valida	Massimo	Saffi	Massimo.Saffi@salgari.com
FJ105AC	2024-02-17	Valida	Massimo	Saffi	Massimo.Saffi@salgari.com
CM929DA	2024-02-03	Scaduta	Marisa	Mastandrea	Marisa.Mastandrea@piatini.com
D0410DG	2024-02-18	Valida	Domenico	Cignaroli	Domenico.Cignaroli@mantegazza.com
FR403FE	2024-02-22	Valida	Baldassare	Pizzamano	Baldassare.Pizzamano@duodo.com
PR375AB	2024-02-14	Scaduta	Maria	Pergolesi	Maria.Pergolesi@brugnaro.com
CK596FE	2024-02-25	Valida	Firenzo	Gagliano	Fiorenzo.Gagliano@fagotto.it
EP391EC	2024-02-02	Scaduta	Gino	Speri	Gino.Speri@dinese.com
ER647EA	2024-02-14	Scaduta	Nina	Goldoni	Nina.Goldoni@binaghi.it
BN214AG	2024-02-12	Scaduta	Sebastiano	Cheda	Sebastiano.Cheda@simondi.it
BK988CB	2024-02-28	Valida	Benvenuto	Gucci	Benvenuto.Gucci@strangio.it
FH845GG	2024-02-25	Valida	Ermes	Gozzi	Ermes.Gozzi@ricci.it
CI402GF	2024-02-17	Valida	Luisa	Moretti	Luisa.Moretti@monduzzi.it
EP402GF	2024-02-03	Scaduta	Ruggero	Gonzaga	Ruggero.Gonzaga@bodonì-navone.it
DP588BE	2024-02-16	Valida	Laureano	Foconi	Laureano.Foconi@trillini-cianciolo.org
GH172DF	2024-02-08	Scaduta	Guido	Carli	Guido.Carli@malpighi.eu
BO716CB	2024-02-28	Valida	Allegra	Lucarelli	Allegra.Lucarelli@anastasi.it
GJ174EA	2024-02-25	Valida	Gustavo	Gabbana	Gustavo.Gabbana@molesini.eu
AL197CB	2024-02-10	Scaduta	Gianpaolo	Magnani	Gianpaolo.Magnani@gaultieri.eu
GQ763FA	2024-02-19	Valida	Maura	Gradenigo	Maura.Gradenigo@saffi.eu

- Visualizza gli autisti che hanno lavorato in una data specifica

```

SELECT a.Nome, a.Cognome, tol.OraInizio, tol.OraFine
FROM Autisti a
JOIN TabellaOrarioLavorativo tol ON a.Matricola = tol.Matricola
WHERE tol.`Data` = "2020-01-02";

```

Nome	Cognome	OraInizio	OraFine
Rembrandt	Ceri	9	17
Tullio	Draghi	9	22
Galasso	Bignardi	9	22
Enzo	Vasari	9	22
Paulina	Iacobucci	9	22
Martina	Tartaglia	10	21
Liana	Gregorio	11	22
Vittorio	Borzomi	11	22

- Visualizza tutti gli autisti che hanno avuto lo stesso turno in una data specifica

```

SELECT a.Nome, a.Cognome, tol.OraInizio, tol.OraFine
FROM Autisti a
JOIN TabellaOrarioLavorativo tol ON a.Matricola = tol.Matricola
WHERE tol.`Data` = "2020-01-02" AND tol.OraInizio = "9" AND tol.OraFine = "17"

```

Nome	Cognome	OraInizio	OraFine
Rembrandt	Ceri	9	17

- Visualizza la somma dei pagamenti effettuati dagli utenti in una data settimana

```

SELECT SUM(tc.Costo) AS Totale FROM TratteCompletate tc
WHERE MONTH (tc.DataRichiesta) = "06" AND DAY (tc.DataRichiesta) BETWEEN 1 AND 7

```

Totale	
16345	

- Visualizza la media dei costi delle tratte che sono state completate a giugno 2023

```

SELECT AVG(tc.Costo) AS MediaCosti, COUNT(*) as NumeroCorse
FROM TratteCompletate tc
WHERE MONTH (tc.DataRichiesta) = "06" AND YEAR(tc.DataRichiesta) = "2023"
ORDER BY MediaCosti

```

MediaCosti	NumeroCorse
54.4828	290

- Visualizza tutte le richieste di manutenzione relative ad uno specifico veicolo

```

SELECT cpg.Motivo, v.* FROM ContattaPerGuasto cpg
JOIN Autisti a ON cpg.Matricola = a.Matricola
JOIN Veicoli v ON a.Matricola = v.Matricola
WHERE v.Targa = "GM903CE";

```

Motivo	Targa	Marca	Modello	NumPosti	Matricola
Spia dell motore accesa	GM903CE	Alfa Romeo	Giulietta	12	825381
Radiatore bucato	GM903CE	Alfa Romeo	Giulietta	12	825381
Batteria scarica	GM903CE	Alfa Romeo	Giulietta	12	825381
Specchietto rotto	GM903CE	Alfa Romeo	Giulietta	12	825381
Rottura degli ammortizzatori	GM903CE	Alfa Romeo	Giulietta	12	825381
La macchina non parte	GM903CE	Alfa Romeo	Giulietta	12	825381
Radiatore bucato	GM903CE	Alfa Romeo	Giulietta	12	825381
Batteria scarica	GM903CE	Alfa Romeo	Giulietta	12	825381
Batteria scarica	GM903CE	Alfa Romeo	Giulietta	12	825381
Specchietto rotto	GM903CE	Alfa Romeo	Giulietta	12	825381
Cambio pasticche dei freni	GM903CE	Alfa Romeo	Giulietta	12	825381

- Visualizza le 10 tratte più gettonate

```
SELECT Partenza, Arrivo, COUNT(*) AS NumeroRichieste
FROM RichiestePrenotazioni rp
GROUP BY Partenza, Arrivo
ORDER BY NumeroRichieste DESC
LIMIT 10;
```

Partenza	Arrivo	NumeroRichieste
Giardinetti	Anagnina	117
Campo de Fiori	Verano	113
Porta Furba	Prima Porta	109
Colosseo	Termini	106
Pigneto	Tufello	104
Prenestina	Tufello	103
Giardinetti	Lucio Sestio	103
Tufello	Campo de Fiori	102
Termini	Tufello	102
Verano	Termini	101

- Visualizza gli utenti che hanno effettuato almeno 5 richieste

```
SELECT u.ID_Utente, u.Nome, u.Cognome, COUNT(*) AS NumeroRichieste
FROM RichiestePrenotazioni rp JOIN Utenti u ON rp.ID_Utente = u.ID_Utente
GROUP BY u.ID_Utente, u.Nome, u.Cognome
HAVING NumeroRichieste >= 5
ORDER BY NumeroRichieste DESC;
```

ID_Utente	Nome	Cognome	NumeroRichieste
1686	Uberto	Filogamo	10
3608	Priscilla	Modugno	10
8120	Ippazio	Muratori	9
5048	Gianna	Luna	8
5412	Giuliano	Bosio	8
6172	Tina	Guinizzelli	8
8257	Achille	Tarchetti	8
9249	Piero	Baglioni	8
16	Adriana	Antonucci	7
1094	Federica	Endrizzi	7
1354	Elvira	Faranda	7
1395	Donna	Seddio	7
1493	Giulio	Boaga	7
1653	Toni	Omma	7
1764	Imelda	Saracino	7
1821	Ninetta	Gramsci	7
2563	Germana	Balotelli	7
2761	Rossana	Dallara	7
2925	Filippa	Zaccardo	7
3037	Ignazio	Zacco	7
3088	Stefania	Carfagna	7
4097	Roberta	Infantino	7
4180	Ezio	Pometta	7
4287	Pierina	Aldobrandi	7
5163	Achille	Trebbi	7
5211	Graziano	Solimena	7
5222	Maria	Tomaselli	7
5470	Luchino	Vergassola	7
5668	Goffredo	Giannelli	7
5729	Gilberto	Schiaparelli	7
6404	Dario	Pulci	7
6966	Ottone	Pisani	7
7676	Maria	Gibilisco	7
8487	Niccolò	Donatoni	7

- Visualizza il motivo di rifiuto delle richieste di prenotazione che occorre più spesso

```
SELECT tr.Motivazione, COUNT(*) AS NumeroOcorrenze
FROM TratteRifiutate tr GROUP BY tr.Motivazione
ORDER BY NumeroOcorrenze DESC
LIMIT 1;
```

Motivazione	NumeroOcorrenze
Indisponibilità al servizio	1024

- Visualizza lo storico dei turni che un dato autista ha effettuato nel corso del tempo

```
SELECT a.Nome,a.Cognome,tol./*
FROM TabellaOrarioLavorativo tol
JOIN Autisti a ON tol.Matricola = a.Matricola
WHERE a.Matricola = "569"
ORDER BY tol.`Data`;
```

Nome	Cognome	Matricola	OraInizio	OraFine	Data
Isa	Manzoni	569	10	21	2020-04-22
Isa	Manzoni	569	11	20	2020-07-30
Isa	Manzoni	569	9	17	2022-01-01

- Visualizza tutte le tratte completate che non hanno un feedback

```
SELECT tc./*
FROM TratteCompletate tc
WHERE (tc.ID_Utente,tc.Partenza,tc.Arrivo,tc.DataRichiesta,tc.OrarioRichiesta) NOT IN
(
    SELECT f.ID_Utente,f.Partenza,f.Arrivo,f.DataRichiesta,f.OrarioRichiesta FROM Feedback
f
);
```

ID_Utente	Partenza	Arrivo	DataRichiesta	OrarioRichiesta	Costo	MetodoDiPagamento	DataPagamento	OraPagamento	Autista
295	Verano	Termini	2020-01-14	21	50	Satispay	2020-01-14	23	716513
1321	Anagnina	Colosseo	2021-03-23	15	50	Carta di credito	2021-03-23	16	756113
1848	Anagnina	Verano	2020-01-02	15	25	CashUp	2020-01-02	23	798512
2773	Trastevere	Prenestina	2022-03-02	16	65	CashUp	2022-03-02	21	573075
3030	Tor Bella Monaca	Termini	2020-03-06	22	115	CashUp	2020-03-06	23	387732
3320	Tor Bella Monaca	Prima Porta	2021-02-10	9	50	Contanti	2021-02-10	23	685649
3327	Prenestina	Prima Porta	2021-05-01	9	50	Satispay	2021-05-01	23	592283
3568	Prenestina	Porta Furba	2022-10-30	22	65	Postepay	2022-10-30	22	992683
3622	Anagnina	Termini	2023-08-18	22	50	Satispay	2023-08-18	23	301394
4375	Porta Furba	Prenestina	2023-08-25	21	50	Satispay	2023-08-25	23	412480
4443	Tufello	Giardinetti	2022-09-28	11	65	CashUp	2022-09-28	23	420543
4701	Prenestina	Pigneto	2022-10-09	11	115	Paypal	2022-10-09	11	512282
5499	Anagnina	Prenestina	2021-09-22	20	65	CashUp	2021-09-22	21	319131
5769	Anagnina	Tor Bella Monaca	2023-01-29	10	25	Satispay	2023-01-29	20	95517
6396	Lucio Sestio	Termini	2023-11-11	16	50	Contanti	2023-11-11	9	878703
6866	Lucio Sestio	Palmo Togliatti	2021-02-16	22	115	Contanti	2021-02-16	23	326308
7064	Colosseo	Termini	2024-01-07	21	35	Postepay	2024-01-07	23	488311
8060	Salaria	Giardinetti	2020-10-20	16	65	Paypal	2020-10-20	20	341709
8120	Colosseo	Lucio Sestio	2021-04-12	9	50	Postepay	2021-04-12	23	220288
8414	Tufello	Pigneto	2023-01-18	10	50	CashUp	2023-01-18	10	828681
9101	Lucio Sestio	Giardinetti	2023-07-24	20	25	CashUp	2023-07-24	23	645976
9226	Anagnina	Salaria	2023-12-06	14	25	Postepay	2023-12-06	14	320627

- Visualizza tutte le richieste di prenotazione effettuate da un determinato utente

```
SELECT rp.* FROM RichiestePrenotazioni rp
JOIN Utenti u ON rp.ID_Utente = u.ID_Utente
WHERE u.Nome = 'Serafina' AND u.Cognome = 'Canali'
```

ID_Utente	Partenza	Arrivo	DataRichiesta	OrarioRichiesta	NumeroPassengeri
181	Campo de Fiori	Porta Furba	2023-01-09	9	1
181	Pigneto	Prenestina	2020-09-07	11	3
181	Tufello	Pigneto	2023-07-15	16	6

- Visualizza la media delle stelle ottenute da un singolo autista

```
SELECT a.Nome, a.Cognome, AVG(f.StelleUtente) AS MediaStelle
FROM Feedback f
JOIN TratteCompletate tc ON (f.ID_Utente,f.Partenza,f.Arrivo,f.DataRichiesta,f.OrarioRichiesta)
= (tc.ID_Utente,tc.Partenza,tc.Arrivo,tc.DataRichiesta,tc.OrarioRichiesta)
JOIN Autisti a ON tc.Autista = a.Matricola
WHERE tc.Autista = '569'
GROUP BY a.Nome, a.Cognome
ORDER BY MediaStelle DESC
```

Nome	Cognome	MediaStelle
Isa	Manzoni	2.3333

- Visualizza il numero totale delle assicurazioni Kasko

```
SELECT COUNT(a.Tipo) AS TotaleKasko
FROM Assicurazioni a
WHERE a.Tipo = 'Kasko';
```

TotaleKasko
986

- Visualizza tutti gli autisti che hanno una certa categoria di patente

```
SELECT a.Nome, a.Cognome, pt.Categoria
FROM Autisti a
JOIN Patenti pt ON a.NumeroPatente = pt.NumeroPatente
WHERE pt.Categoria = "B96"
```

Nome	Cognome	Categoria
Maura	Berlusconi	B96
Emma	Morlacchi	B96
Roberta	Gigli	B96
Giacinto	Jacuzzi	B96
Gino	Cuomo	B96
Giacobbe	Bramante	B96
Raffaella	Fallaci	B96
Giuliano	Gualandi	B96
Alessandro	Gilardoni	B96
Alina	Borrani	B96
Camilla	Faugno	B96
Annunziata	Anguillara	B96
Torquato	Tosi	B96
Pierluigi	Poerio	B96
Nino	Vismara	B96
Imelda	Surian	B96
Antonino	Gargallo	B96
Orazio	Impastato	B96
Guglielmo	Lucarelli	B96
Ivo	Basso	B96
Licia	Navarria	B96
Lamberto	Foà	B96
Giustino	Palladio	B96
Tatiana	Iannotti	B96
Alphons	Ruberto	B96
Amleto	Broggini	B96
Livio	Sforza	B96
Graziano	Bonino	B96
Pina	Petralli	B96
Gabriele	Romagnoli	B96
Paloma	Cavalcanti	B96
Niccolò	Leoncavallo	B96
Filippa	Persico	B96
Enzo	Vigorelli	B96

- Visualizza il numero di feedback con almeno 3 stelle lasciati da ogni utente

```

SELECT u.Nome,u.Cognome, COUNT(*) AS NumeroFeedback3Stelle
FROM Feedback f JOIN TratteCompletate tc ON
(f.ID_Utente,f.Partenza,f.Arrivo,f.DataRichiesta,f.OrarioRichiesta) =
(tc.ID_Utente,tc.Partenza,tc.Arrivo,tc.DataRichiesta,tc.OrarioRichiesta)
JOIN Utenti u ON tc.ID_Utente = u.ID_Utente
WHERE f.StelleUtente >= 3
GROUP BY u.Nome,u.Cognome
ORDER BY NumeroFeedback3Stelle DESC

```

Nome	Cognome	NumeroFeedback3Stelle
Gilberto	Schiaparelli	7
Stefania	Carfagna	6
Ruggero	Castioni	5
Pierangelo	Barozzi	5
Amedeo	Bossi	5
Niccolò	Donatoni	5
Tonino	Telesio	5
Piero	Barillaro	5
Leone	Chittolini	5
Azeglio	Pometta	5
Gianna	Luna	5
Imelda	Saracino	5
Fabia	Tonisto	5
Maria	Borgia	5
Uberto	Filogamo	5
Pierpaolo	Casini	5
Aurora	Spallanzani	5
Rodolfo	Battelli	5
Alessandra	Lollobrigida	5
Giulia	Tagliafierro	4
Silvio	Battisti	4
Fiorenzo	Zichichi	4
Cecilia	Renier	4
Arsenio	Onio	4
Maria	Molesini	4
Donatello	Sermonti	4
Fabia	Doria	4
Geronimo	Novaro	4
Ernesto	Fibonacci	4
Beatrice	Mazzini	4
Costanzo	Montesano	4
Cristina	Taliani	4
Renzo	Romano	4
Allegra	Trombetta	4

- Visualizza l'ultima richiesta di prenotazione di un certo utente, aggiungendo (**solo in output**) un campo che dice se la Richiesta fa parte di una tratta completata o no

```

SELECT rp.*,
IF((rp.ID_Utente, rp.Partenza, rp.Arrivo, rp.DataRichiesta, rp.OrarioRichiesta) IN (SELECT
tc.ID_Utente, tc.Partenza, tc.Arrivo, tc.DataRichiesta, tc.OrarioRichiesta FROM TratteCompletate
tc), 'SI', 'NO' ) AS Completata
FROM RichiestePrenotazioni rp
WHERE rp.ID_Utente = '738' AND rp.DataRichiesta
IN
(
    SELECT MAX(DataRichiesta)
    FROM RichiestePrenotazioni
    WHERE ID_Utente = '738'
)

```

ID_Utente	Partenza	Arrivo	DataRichiesta	OrarioRichiesta	NumeroPasseggeri	Completata
738	Campo de Fiori Tor Bella Monaca		2021-12-02	10	6	SI

- Trova tutti gli autisti che non hanno mai effettuato una richiesta di manutenzione

```

SELECT Matricola, Nome, Cognome, Email, NumeroTelefono, Stipendio
FROM Autisti A
WHERE A.ID_Autista NOT IN
(
    SELECT cpg.ID_Autista FROM ContattaPerGuasto cpg
);

```

Matricola	Nome	Cognome	Email	NumeroTelefono	Stipendio
15	Francesco	Salvini	Francesco.Salvini@lucciano.it	3311146210	1100
646	Tiziano	Toso	Tiziano.Toso@columbo.com	+39 041061265	800
700	Teresa	Beccaria	Teresa.Beccaria@uggieri.it	3371497746	800
718	Emma	Galilei	Emma.Galilei@bottigliero-montecchi.com	0371828902	1200
722	Giacomo	Mercadante	Giacomo.Mercadante@pennetta-jovinelli.eu	+39 3733911312	1100
894	Ronaldo	Treves	Ronaldo.Treves@roncalli-morosini.net	+39 0331469933	800
1029	Jolanda	Rastelli	Jolanda.Rastelli@calarco.com	0356005219	1100
1327	Ludovica	Celentano	Ludovica.Celentano@orlando-andreotti.it	0592329174	1100
1487	Lorenzo	Treccani	Lorenzo.Treccani@ienzo-trombetta.it	377611725	900
1769	Dante	Tosi	Dante.Tosi@mozart.com	3828703864	1200
2044	Umberto	Anastasi	Umberto.Anastasi@rensi.com	+39 350075264	1200
2447	Alberto	Vitturi	Alberto.Vitturi@calgari-trebbi.com	+39 0183818726	1200
2922	Bernardo	Pinamonte	Bernardo.Pinamonte@scarpa.net	3793692330	800
3097	Fedele	Vergerio	Fedele.Vergerio@finzi.com	+39 095813157	1100
3450	Vincentio	Bruscantini	Vincentio.Bruscantini@versace.com	+39 0363854337	900
4566	Jolanda	Varano	Jolanda.Varano@catenazzi-pertini.it	0994829424	1100
5092	Gionata	Boccioni	Gionata.Boccioni@ammaniti.it	+39 0933473348	1100
5093	Benito	Badoer	Benito.Badoer@osiglia-mercadante.net	072168899	900
5534	Martina	Tartaglia	Martina.Tartaglia@iannotti-chiappetta.it	+39 0322656661	900
5857	Biagio	Gabba	Biagio.Gabba@zanzi.com	+39 375569853	1200
6233	Camillo	Ceravolo	Camillo.Ceravolo@pugliese-carocci.com	04315306067	900
6993	Leone	Battisti	Leone.Battisti@bonino.it	+39 351946728	900
7174	Silvio	Ginesio	Silvio.Ginesio@muratori.eu	+39 351227624	1200
7755	Torquato	Venier	Torquato.Venier@praga.it	3349813638	900
7793	Eraldo	Sbarbaro	Eraldo.Sbarbaro@tasso-petrassi.com	+39 032115295	1200
7827	Beppe	Bevilacqua	Beppe.Bevilacqua@iannuzzi-cignaroli.com	+39 373156547	1100
8750	Paola	Mengolo	Paola.Mengolo@vianello-nibali.eu	+39 086177752	900
9150	Eraldo	Cagnin	Eraldo.Cagnin@panatta.com	+39 0825840221	1100
9316	Nicoletta	Tonisto	Nicoletta.Tonisto@trincavelli.eu	37837599803	1200
10093	Federica	Bartoli	Federica.Bartoli@capone-turci.it	34329122078	800
10452	Vanessa	Campano	Vanessa.Campano@tozzo.it	07513710823	1100
10887	Serafina	Giusti	Serafina.Giusti@peranda.it	+39 351211548	900
11885	Silvestro	Gaggini	Silvestro.Gaggini@pergolesi.com	+39 056689599	1100
11992	Mirko	Venier	Mirko.Venier@fusani.it	09315146955	1200

- Visualizza il totale dei pagamenti relativi ad un determinato giorno

```
SELECT SUM(tc.Costo) AS TotalePagamenti
FROM TratteCompletate tc
WHERE tc.DataRichiesta = "2023-06-05"
```

TotalePagamenti
220

- Visualizza l'estratto conto generale dei 5 utenti che speso di più

```
SELECT sum(tc.Costo) as SommaTotale, tc.ID_Utente
FROM TratteCompletate tc
GROUP BY tc.ID_Utente
ORDER BY SommaTotale DESC
LIMIT 5
```

SommaTotale	ID_Utente
635	8120
550	8624
525	6313
505	8487
505	1094

- Visualizza gli autisti con lo stipendio più alto

```

SELECT *
FROM Autisti
WHERE Stipendio =
(
    SELECT MAX(Stipendio)
    FROM Autisti
);

```

Matricola	Nome	Cognome	Email	DDN	NumeroTelefono	NumeroPatente	Stipendio
718	Emma	Galilei	Emma.Galilei@bottiglieri-montecchi.com	1986-12-17	0371828902	B0K15S0FF	1200
1769	Dante	Tosi	Dante.Tosi@mozart.com	1985-07-08	3828703864	NJYHMYL2Q	1200
2044	Umberto	Anastasi	Umberto.Anastasi@ensi.com	1982-01-23	+39 350075264	W0PF5R5LT	1200
2447	Alberto	Vitturi	Alberto.Vitturi@calgari-trebbi.com	1998-09-09	+39 0183818726	L171E10AM	1200
5857	Biagio	Gabba	Biagio.Gabba@zanzi.com	1979-01-04	+39 375569853	SUIWCPIWX	1200
7174	Silvio	Ginesio	Silvio.Ginesio@muratori.eu	1985-05-07	+39 351227624	R07FMYLYZ	1200
7793	Eraldo	Sbarbaro	Eraldo.Sbarbaro@tasso-petrossi.com	1980-07-14	+39 032115295	62D89BFNG	1200
9316	Nicoletta	Tonisto	Nicoletta.Tonisto@trincavelli.eu	1976-08-15	37837599803	FEVN602IN	1200
11992	Mirko	Venier	Mirko.Venier@fusani.it	1984-03-05	0915146955	PLOCNOU2G	1200
12346	Ornella	Prodi	Ornella.Prodi@gabba.com	1993-07-28	+39 09193842508	I7127M9ZD	1200
12750	Priscilla	Micheletti	Priscilla.Micheletti@scotti.it	1996-11-18	04459245370	ISUTQFT4U	1200
13086	Marcella	Bacosi	Marcella.Bacosi@morpurgo.it	1982-07-20	0961929827	VP50R601K	1200
13643	Luciano	Civaschi	Luciano.Civaschi@toscani.eu	1998-08-25	079100893	KUMCOCRK3	1200
13752	Gian	Michelangeli	Gian.Michelangeli@ossiga-bianchini.com	2000-10-08	058646799	YCWSUUCLH	1200
14763	Enzo	Vigorelli	Enzo.Vigorelli@stefanelli-padovano.com	1994-06-04	+39 0434452177	10D7J6LB3	1200
15359	Priscilla	Folliero	Priscilla.Folliero@barsanti-galilei.net	1980-03-02	35575489716	SESUU3NNB	1200
15735	Giuliano	Stradivari	Giuliano.Stradivari@gelli-turrini.it	1983-03-20	+39 073317507	RIHXTUNGI	1200
15875	Benvenuto	Zampa	Benvenuto.Zampa@gritti.net	1997-01-07	+39 0355958335	J8FJZ5KNI	1200
16945	Romeo	Taliercio	Romeo.Taliercio@franceschi.it	1989-04-26	+39 3886153786	9KEG4W7EO	1200
18503	Michelotto	Nicolini	Michelotto.Nicolini@cattaneo.com	1985-04-13	+39 3484835782	UBTRS9CCU	1200
20149	Ilaria	Einaudi	Ilaria.Einaudi@foa.it	1999-03-15	+39 0532122309	MNPUS05PD	1200
21098	Hugo	Cainero	Hugo.Cainero@magnani.org	1976-06-03	3977237147	L0NQR20Z2	1200
21462	Gian	Platini	Gian.Platini@finetti.it	1991-10-26	0984349178	NACBA58LE	1200
22222	Maurilio	Serao	Maurilio.Serao@rosmini.com	1985-03-28	04211154592	OKIPJPLSN	1200
25709	Caterina	Marsili	Caterina.Marsili@carpaccioantonioni.com	2000-06-03	+39 05740087538	X18FKM9VZ	1200
26873	Giampiero	Burcardo	Giampiero.Burcardo@interminelli.it	1998-04-09	03443135091	QF64RV18K	1200
27299	Gianfranco	Boccherini	Gianfranco.Boccherini@deledda-scotto.net	1980-05-04	034493693	IF2B48QWU	1200
27335	Marcello	Ginese	Marcello.Ginese@carli.it	1997-03-25	+39 3777111083	MTY4I38FF	1200
27499	Ruggero	Manzoni	Ruggero.Manzoni@giannetti.it	1977-04-19	0721664713	CVM90TA6S	1200
27687	Milena	Errigo	Milena.Errigo@gulotta-betttoni.it	1978-12-15	05880397682	6YU77UAE6	1200
28059	Damiano	Mastandrea	Damiano.Mastandrea@ienzo-giradello.org	1977-03-22	+39 3757773411	6FLJJ9610	1200
28937	Milena	Tamburello	Milena.Tamburello@martucci.it	1991-05-27	+39 073744087	LF9KVX8LM	1200
29594	Lolita	Emo	Lolita.Emo@bianchini-farinelli.eu	1985-08-31	323993721	M8YA04C22	1200

- Trova gli autisti che hanno completato il minor numero di corse in un determinato giorno

```

SELECT a.Matricola ,a.Nome,a.Cognome, COUNT(*) AS NumeroCorseEffettuate
FROM TratteCompletate tc
JOIN Autisti a ON tc.Autista = a.Matricola
WHERE tc.DataRichiesta = "2023-06-05"
GROUP BY Matricola ,Nome ,Cognome
ORDER BY NumeroCorseEffettuate

```

Matricola	Nome	Cognome	NumeroCorseEffettuate
498288	Liberto	Tuzzolino	1
508042	Elvira	Pisano	1
554330	Alberico	Calvo	1
673395	Rosina	Antelami	1
901634	Piermaria	Vivaldi	1
965611	Pietro	Speri	1

- Visualizza tutti i dati di un determinato utente, comprese le carte a lui associate

```

SELECT u.* , c.NumeroCarta, c.DataScadenza, c.CVV
FROM Utenti u JOIN Carta c ON c.ID_Utente = u.ID_Utente
WHERE Nome = "Rosa" AND Cognome = "Lussu"

```

ID_Utente	Nome	Cognome	Email	Password	DDN	NumeroCarta	DataScadenza	CVV
723	Rosa	Lussu	Rosa.Lussu@panzera.org	*****	1992-11-09	4268 4638 0431 3922	2031-08-18	***
723	Rosa	Lussu	Rosa.Lussu@panzera.org	*****	1992-11-09	4281 3226 8418 7502	2031-03-20	***

- Visualizza tutti gli utenti che hanno almeno 2 carte associate

```
SELECT u.ID_Utente, u.Nome, u.Cognome, COUNT(*) AS NumeroCarteAssociate
FROM Utenti u JOIN Carta c ON c.ID_Utente = u.ID_Utente
GROUP BY u.ID_Utente, u.Nome, u.Cognome
HAVING NumeroCarteAssociate > 2
ORDER BY NumeroCarteAssociate DESC
```

ID_Utente	Nome	Cognome	NumeroCarteAssociate
5969	Rodolfo	Satta	12
1271	Lauretta	Tonisto	9
5387	Arnulfo	Cuda	9
2162	Mirco	Pace	8
3078	Giustino	Emanuelli	8
4234	Ciro	Pozzecco	8
4792	Emilio	Ferretti	8
4967	Pasquale	Bonaventura	8
6895	Fedele	Fattori	8
7374	Laura	Marenzio	8
7377	Telemaco	Duse	8
7521	Graziella	Zoppetto	8
7965	Azeglio	Agostinelli	8
8648	Gianmarco	Comisso	8
8744	Annetta	Veneziano	8
478	Rodolfo	Alonzi	7
1422	Lucia	Valguarnera	7
1457	Danilo	Fantini	7
1541	Dino	Rosiello	7
1973	Simonetta	Gadda	7
2022	Raffaella	Gualtieri	7
2379	Licia	Calgari	7
2454	Amico	Cattaneo	7
2645	Leone	Bignardi	7
3137	Teresa	Sabbatini	7
3326	Gianpaolo	Salvini	7
4485	Camilla	Druso	7
4538	Stefani	Sraffa	7
4613	Micheletto	Salvo	7
5115	Flavio	Giovannotti	7
5442	Girolamo	Piovani	7
5457	Flavio	Tasso	7
5869	Gian	Galilei	7

Ottimizzazione

Di seguito abbiamo selezionato degli attributi su cui creare indici secondari per velocizzare l'esecuzione delle query. Ovviamente, non abbiamo creato troppi indici per una questione di costi di memoria.

Gli indici occupano memoria e quindi abbiamo trovato un compromesso, creando indici solo per gli attributi più richiesti.

Creazione di indici in MySQL

```
CREATE INDEX idx_name
ON Autisti(Nome);
```

```
CREATE INDEX idx_part
ON Feedback(Partenza);
```

```
CREATE INDEX idx_ut_name
ON Utenti(Nome);
```

```
CREATE INDEX idx_data
ON TratteCompletate(DataRichiesta);
```

Utilizzando questi indici secondari, abbiamo la versione ottimizzata di alcune delle query descritte in precedenza, che vengono eseguite sui dati casuali generati dal programma Python. Riportiamo inoltre, la frazione di miglioramento temporale delle ottimizzazioni.

Formula di miglioramento : $100 * (\text{originale}-\text{nuovo}) / \text{originale}$

Visualizza gli autisti con lo stipendio più alto

```
SELECT *
FROM Autisti
WHERE Stipendio =
(
    SELECT MAX(Stipendio)
    FROM Autisti
);
```

Prima della creazione dell'index sul campo "Nome" di Autisti, il tempo di esecuzione della query è il seguente

```
703 rows in set
Time: 0.167s
```

Dopo aver creato l'index, il tempo di esecuzione è il seguente:

```
703 rows in set
Time: 0.105s
```

La formula di miglioramento risulta essere la seguente

$$100 * \frac{(0.167 - 0.105)}{0.167} \sim 37,12\%$$

Quindi, volendo arrotondare, abbiamo un miglioramento di circa il 38%

Visualizza il numero di feedback con almeno 3 stelle lasciati da ogni utente

```
SELECT u.Nome, u.Cognome, COUNT(*) AS NumeroFeedback3Stelle
FROM Feedback f JOIN TratteCompletate tc ON
(f.ID_Utente, f.Partenza, f.Arrivo, f.DataRichiesta, f.OrarioRichiesta) =
(tc.ID_Utente, tc.Partenza, tc.Arrivo, tc.DataRichiesta, tc.OrarioRichiesta)
JOIN Utenti u ON tc.ID_Utente = u.ID_Utente
WHERE f.StelleUtente >= 3
GROUP BY u.Nome, u.Cognome
ORDER BY NumeroFeedback3Stelle DESC
```

Prima di aver creato l'index sul campo "Partenza" della tabella Feedback, il tempo di esecuzione della query è il seguente

```
5924 rows in set
Time: 0.388s
```

Dopo aver creato l'index, il tempo di esecuzione risulta essere:

```
5924 rows in set
Time: 0.334s
```

La formula di miglioramento risulta essere la seguente

$$100 * \frac{(0.388 - 0.334)}{0.388} \sim 13,91\%$$

Quindi, volendo arrotondare, avremmo un miglioramento del 20%

Visualizza gli utenti che hanno effettuato almeno 5 richieste

```

SELECT u.ID_Utente, u.Nome, u.Cognome, COUNT(*) AS NumeroRichieste
FROM RichiestePrenotazioni rp JOIN Utenti u ON rp.ID_Utente = u.ID_Utente
GROUP BY u.ID_Utente, u.Nome, u.Cognome
HAVING NumeroRichieste >= 5
ORDER BY NumeroRichieste DESC;

```

Prima di aver creato l'index sul campo "Nome" sulla tabella Utenti, il tempo di esecuzione della query è il seguente

```

540 rows in set
Time: 0.161s

```

Dopo aver creato l'index, il tempo di esecuzione risulta essere:

```

540 rows in set
Time: 0.124s

```

La formula di miglioramento risulta essere la seguente

$$100 * \frac{(0.161 - 0.124)}{0.161} \sim 22.98 \%$$

Quindi, volendo arrotondare, avremmo un miglioramento del 23%

Trova gli autisti che hanno completato il minor numero di corse in un determinato giorno

```

SELECT a.Matricola ,a.Nome,a.Cognome, COUNT(*) AS NumeroCorseEffettuate
FROM TratteCompletate tc
JOIN Autisti a ON tc.Autista = a.Matricola
WHERE tc.DataRichiesta = "2023-06-05"
GROUP BY Matricola ,Nome ,Cognome
ORDER BY NumeroCorseEffettuate

```

Prima di aver creato l'index sul campo "DataRichiesta", dell'entità TratteCompletate, il tempo di esecuzione della query è il seguente

```

6 rows in set
Time: 0.019s

```

Dopo aver creato l'index, il tempo di esecuzione risulta essere:

```

6 rows in set
Time: 0.010s

```

La formula di miglioramento risulta essere la seguente

$$100 * \frac{0.019 - 0.010}{0.019} \sim 47.36 \%$$

Quindi, volendo arrotondare, avremmo un miglioramento del 48%

Algebra Relazionale

L'algebra relazionale è un linguaggio query *procedurale* in notazione algebrica. In una query, si applicano sequenzialmente le operazioni alle relazioni. Ogni operazione (unaria o binaria) riceve in input una relazione e ne produce un'altra in output.

Le operazioni **primitive** sono:

- Selezione (σ)
- Proiezione (π)

- Unione (\cup)
- Differenza Insiemistica ($-$)
- Prodotto Cartesiano (X)
- Ridenominazione (ρ)

Esistono altre operazioni da esse derivabili, tra cui l'intersezione insiemistica (\cap).

Di seguito troviamo alcune query sul nostro database scritte in Algebra Relazionale:

Visualizza tutte le tratte completate che non hanno un feedback

```
SELECT tc.* FROM TratteCompletate tc
WHERE tc.ID_TrattaC NOT IN
(
    SELECT ID_TrattaCompletata FROM Feedback f
);
```

In algebra relazionale la query diventa

$$pk = tc.ID_Utente, tc.Partenza, tc.Arrivo, tc.DataRichiesta, tc.OraRichiesta \\ \pi_{tc,pk}(TratteCompletate \bowtie (\pi_{tc,pk}(TratteCompletate) - \pi_{tc,pk}(Feedback)))$$

Dove:

- π rappresenta l'operazione di proiezione.
- σ rappresenta l'operazione di selezione.
- \bowtie rappresenta l'operazione di join.

Per questioni di semplicità, abbiamo denominato con pk tutta la chiave primaria dell'entità TratteCompletate, ovvero la composizione dei campi ID_Utente, Partenza,Arrivo,DataRichiesta,OraRichiesta

Visualizza tutti i dati di un determinato utente, comprese le carte a lui associate

```
SELECT u.* , c.NumeroCarta, c.DataScadenza, c.CVV
FROM Utenti u JOIN Carta c ON c.ID_Utente = u.ID_Utente
WHERE Nome = "Rosa" AND Cognome = "Lussu"
```

In algebra relazionale la query diventa:

$$\begin{aligned} & \text{Utenti} \bowtie_{ID_Utente=ID_Utente} \text{Carta} = A \\ & \sigma_{\text{Nome}='Rosa', \text{Cognome}='Lussu'}(A) \end{aligned}$$

Dove:

- π rappresenta l'operazione di proiezione.
- σ rappresenta l'operazione di selezione.
- \bowtie rappresenta l'operazione di join.

L'operazione di join (\bowtie) viene eseguita sulla condizione $ID_Utente=ID_Utente$, e successivamente vengono selezionate le righe in cui $\text{Nome}=\text{"Rosa"}$ e $\text{Cognome}=\text{"Lussu"}$, dopodiché viene applicata la proiezione sui campi specificati.

Per questioni di semplicità, abbiamo denominato con A tutta la parte del join

Visualizza tutti i veicoli la cui assicurazione scadrà entro febbraio 2024

```
SELECT a.Targa, a.DDS AS DataScadenza, a.Stato, a2.Nome, a2.Cognome, a2.Email
FROM Assicurazioni a
JOIN Veicoli v ON v.Targa = a.Targa
```

```

JOIN Autisti a2 ON v.Matricola = a2.Matricola
WHERE YEAR(a.DDS) = "2024" AND MONTH(a.DDS) = "02";

```

In algebra relazionale la query diventa:

$$\text{Assicurazioni } a \bowtie_{v.Targa=a.Targa} \text{Veicoli } v \bowtie_{v.Matricola=a2.Matricola} \text{Autisti } a2 = A \\ \pi_{a.Targa,a.DataScadenza,a.Stato,a2.Nome,a2.Cognome,a2.Email}(\sigma_{\text{DataScadenza} < '2024-02-01'}(A))$$

Visualizza tutti gli autisti che hanno una certa categoria di patente

```

SELECT a.Nome, a.Cognome, pt.Categoria
FROM Autisti a
JOIN Patenti pt ON a.NumeroPatente = pt.NumeroPatente
WHERE pt.Categoria = "B96"

```

In algebra relazionale diventa:

$$\text{Autisti} \bowtie_{\text{NumeroPatente}=\text{NumeroPatente}} \text{Patenti} = A \\ \pi_{\text{Nome},\text{Cognome},\text{Categoria}}(\sigma_{\text{Categoria}='B96'}(A))$$

Trova tutti gli autisti che non hanno mai effettuato una richiesta di manutenzione

```

SELECT A.*
FROM Autisti A
WHERE A.Matricola NOT IN
(
    SELECT cpg.Matricola FROM ContattaPerGuasto cpg
);

```

In algebra relazionale la query diventa

$$\pi_{A.*}(\text{Autisti} \bowtie (\pi_{\text{Matricola}}(\text{Autisti}) - \pi_{\text{Matricola}}(\text{ContattaPerGuasto})))$$

Calcolo Relazionale

Il calcolo relazionale è un linguaggio query non procedurale ma *dichiarativo*. Invece dell'algebra, utilizza il calcolo dei predicati matematici del primo ordine in notazione logica. L'output di una query è una relazione che contiene solo tuple che soddisfano le formule logiche espresse. Il potere espressivo del calcolo relazionale è dunque equivalente a quello dell'algebra relazionale.

Versioni:

1. Calcolo relazionale sui domini
2. *Calcolo relazionale sulle tuple con dichiarazione di range*

Di seguito sono alcune query espresse tramite il *calcolo relazionale sulle tuple con dichiarazione di range*:

Visualizza tutte le tratte complete che non hanno un feedback

$$pk = tc.ID_Utente,tc.Partenza,tc.Arrivo,tc.DataRichiesta,tc.OraRichiesta \\ p = tc.pk \in tc \wedge tc.pk \notin f \\ \{tc.* \mid tc(\text{TratteComplete}),f(\text{Feedback}) \mid p\}$$

Per questioni di semplicità, abbiamo denominato con `pk` tutta la chiave primaria dell'entità `TratteComplete`, ovvero la composizione dei campi `ID_Utente`, `Partenza`, `Arrivo`, `DataRichiesta`, `OraRichiesta`

Visualizza tutti i dati di un determinato utente, comprese le carte a lui associate

$$p = \{(u.Nome='Rosa' \wedge u.Cognome='Lussu') \wedge (c.ID_Utente=u.ID_Utente)\} \\ \{u.* , c.(NumeroCarta, DataScadenza, CVV) \mid u(\text{Utenti}), c(\text{Carta}) \mid p\}$$

Visualizza tutti i veicoli la cui assicurazione scadrà entro febbraio 2024

```
p = {(a.DataScadenza < '2024-01-02') ∧ (v.Targa=a.Targa) ∧ (a2.Matricola=v.Matricola)}
{a.(Targa,DataScadenza,Stato),a2.(Nome,Cognome,Email) | a(Assicurazione),a2(Autisti) | p}
```

Visualizza tutti gli autisti che hanno una certa categoria di patente

```
p = {(pt.Categoria='B96' ∧ a.NumeroPatente = pt.NumeroPatente)}
{a.(Nome,Cognome),pt.(Categoria) | pt(Patenti), a(Autisiti) | p}
```

Trova tutti gli autisti che non hanno mai effettuato una richiesta di manutenzione

```
p = {a.Matrixola ∈ a ∧ a.Matricola ∉ cpg}
{a.* | a(Autista), cpg(ContattaPerGuasto) | p}
```

Sicurezza

Ovviamente in un database aziendale devono essere presenti diverse tipologie di utenti con diversi diritti, nella nostra modellizzazione della realtà, infatti, abbiamo definito 2 classi di utenti:

- un amministratore che ha tutti i diritti
- gli autisti, gli addetti marketing e i manutentori che possono aggiungere righe e fare query

Inoltre, si è definito un terzo utente che ha accesso solamente a delle view in modalità lettura, questo perché non gli si vuole dare accesso alle tabelle originali per questioni di sicurezza. Ovviamente la creazione di questo ultimo utente ha il solo fine dimostrativo e non sarebbe effettivamente inserito in un progetto reale.

Le view sono tabelle che non memorizzano dati, esse condividono lo stesso spazio delle tabelle originali. Spesso vengono assegnate ad altri utenti con specifici campi oscurati anche se il loro utilizzo inappropriate può portare all'inconsistenza del database.

Views

Visualizza tutti gli utenti che hanno almeno 2 carte associate

```
CREATE VIEW CartePerUtente AS
(
    SELECT u.ID_Utente, u.Nome, u.Cognome, COUNT(*) AS NumeroCarteAssociate
    FROM Utenti u JOIN Carta c ON c.ID_Utente = u.ID_Utente
    GROUP BY u.ID_Utente, u.Nome, u.Cognome
    HAVING NumeroCarteAssociate > 2
    ORDER BY NumeroCarteAssociate DESC
)
```

```
MySQL lfn@160.80.216.209:VroomA> CREATE VIEW CartePerUtente AS
-> (
->     SELECT u.ID_Utente, u.Nome, u.Cognome, COUNT(*) AS NumeroCarteAssociate
->     FROM Utenti u JOIN Carta c ON c.ID_Utente = u.ID_Utente
->     GROUP BY u.ID_Utente, u.Nome, u.Cognome
->     HAVING NumeroCarteAssociate > 2
->     ORDER BY NumeroCarteAssociate DESC
-> )
Query OK, 0 rows affected
Time: 0.011s
```

L'esecuzione della view darà il seguente risultato

ID_Utente	Nome	Cognome	NumeroCarteAssociate
5969	Rodolfo	Satta	12
1271	Lauretta	Tonisto	9
5387	Arnulfo	Cuda	9
2162	Mirco	Pace	8
3078	Giustino	Emanuelli	8
4234	Ciro	Pozzecco	8
4792	Emilio	Ferretti	8
4967	Pasquale	Bonaventura	8
6895	Fedele	Fattori	8
7374	Laura	Marenzio	8
7377	Telemaco	Duse	8
7521	Graziella	Zoppetto	8
7965	Azeglio	Agostinelli	8
8648	Gianmarco	Comisso	8
8744	Annetta	Veneziano	8
478	Rodolfo	Alonzi	7
1422	Lucia	Valguarnera	7
1457	Danilo	Fantini	7
1541	Dino	Rosiello	7
1973	Simonetta	Gadda	7
2022	Raffaella	Gualtieri	7
2379	Licia	Calgari	7
2454	Amico	Cattaneo	7
2645	Leone	Bignardi	7
3137	Teresa	Sabbatini	7
3326	Gianpaolo	Salvini	7
4485	Camilla	Druso	7
4538	Stefani	Sraffa	7
4613	Micheletto	Salvo	7

Visualizza il numero di feedback con almeno 3 stelle lasciati da ogni utente

```
CREATE VIEW NumeroFeedbackTreStelle AS
(
    SELECT u.Nome,u.Cognome, COUNT(*) AS NumeroFeedback3Stelle
    FROM Feedback f JOIN TratteCompletate tc ON
(f.ID_Utente,f.Partenza,f.Arrivo,f.DataRichiesta,f.OrarioRichiesta) =
(tc.ID_Utente,tc.Partenza,tc.Arrivo,tc.DataRichiesta,tc.OrarioRichiesta)
    JOIN Utenti u ON tc.ID_Utente = u.ID_Utente
    WHERE f.StelleUtente >= 3
    GROUP BY u.Nome,u.Cognome
    ORDER BY NumeroFeedback3Stelle DESC
)
```

```
MySQL lfn@160.80.216.209:VroomA> CREATE VIEW NumeroFeedbackTreStelle AS
-> (
->     SELECT u.Nome,u.Cognome, COUNT(*) AS NumeroFeedback3Stelle
->     FROM Feedback f JOIN TratteCompletate tc ON (f.ID_Utente,f.Partenza,f.Arrivo,f.DataRichiesta,f.OrarioRichiesta) = (tc.ID_Utente,tc.Partenza,tc.Arrivo,tc.DataRichiesta,tc.OrarioRichiesta)
->     JOIN Utenti u ON tc.ID_Utente = u.ID_Utente
->     WHERE f.StelleUtente >= 3
->     GROUP BY u.Nome,u.Cognome ORDER BY NumeroFeedback3Stelle DESC
-> )
```

Query OK, 0 rows affected
Time: 0.009s

L'esecuzione della view darà il seguente risultato

Nome	Cognome	NumeroFeedback3Stelle
Gilberto	Schiaparelli	7
Stefania	Carfagna	6
Ruggero	Castioni	5
Pierangelo	Barozzi	5
Amedeo	Bossi	5
Niccolò	Donatoni	5
Tonino	Telesio	5
Piero	Barillaro	5
Leone	Chittolini	5
Azeglio	Pometta	5
Gianna	Luna	5
Imelda	Saracino	5
Fabia	Tonisto	5
Maria	Borgia	5
Uberto	Filogamo	5
Pierpaolo	Casini	5
Aurora	Spallanzani	5
Rodolfo	Battelli	5
Alessandra	Lollobrigida	5
Giulia	Tagliafierro	4
Silvio	Battisti	4
Fiorenzo	Zichichi	4
Cecilia	Renier	4

Creazione Utenti

Poiché il progetto rappresenta una realtà aziendale di una società, abbiamo creato 3 classi di utenti in ordine decrescente di grado di privilegi. Un amministratore è colui che gestisce il database e quindi ha tutti i diritti. Il personale ha il diritto di inserire nuove tuple e di effettuare query ai fini lavorativi.

Infine, abbiamo creato anche un generico utente autorizzato solo ad interrogare le view esistenti.

1. **Amministratore:** ha tutti i diritti

```
CREATE USER 'administrator'@'localhost' IDENTIFIED BY 'adminpassword';
REVOKE ALL PRIVILEGES, GRANT OPTION FROM 'administrator'@'localhost';
GRANT ALL ON VroomA.* TO 'administrator'@'localhost';
```

2. **Personale:** Può fare query e inserire tuple

```
CREATE USER 'personale'@'localhost' IDENTIFIED BY 'perspassword';
REVOKE ALL PRIVILEGES, GRANT OPTION FROM 'personale'@'localhost';
GRANT SELECT ON VroomA.* TO 'personale'@'localhost';
GRANT INSERT ON VroomA.* TO 'personale'@'localhost';
```

3. Creazione di un utente che ha il solo diritto di eseguire le view sopra scritte

```
CREATE USER 'lfn'@'localhost' IDENTIFIED BY 'lfnpassword';
REVOKE ALL PRIVILEGES, GRANT OPTION FROM 'lfn'@'localhost';
GRANT SELECT ON CartePerUtente TO 'lfn'@'localhost';
GRANT SELECT ON NumeroFeedbackTreStelle TO 'lfn'@'localhost';
```