

PROGETTO BASI DI DATI

CyberAuto

Moretto Nicholas

Toso Giacomo

A.a. 2021/2022

1. Abstract

CyberAuto è un database per la gestione di una concessionaria. Il sistema deve gestire le ordinazioni da parte dei clienti e prevede tre modalità di acquisto dell'auto. L'acquisto vero e proprio, con possibilità di effettuare un finanziamento, il noleggio e il leasing. Il sistema prevede l'ordine delle auto tramite il sito internet: per questo il cliente deve fornire username e password, oltre che il nome.

Al termine dell'ordine, il cliente ha la possibilità di lasciare una valutazione sulla qualità del servizio offerto.

Ogni auto può essere presente solo in una delle sedi della concessionaria. Per ogni sede, inoltre, c'è un manager che la gestisce, dei venditori e degli impiegati.

Il database gestisce anche il conto della concessionaria, che riguarda gli incassi dovuti alla vendita, al noleggio o ai contratti stipulati di auto e le varie spese degli stipendi del personale e dell'acquisto delle auto stesse.

2. Analisi dei requisiti

2.1. Descrizione dei requisiti

Il progetto vuole rappresentare un database sulla gestione di una concessionaria, della vendita/acquisto di auto e dei pagamenti.

Il cliente viene identificato da un ID univoco, e dovrà registrarsi con un nome utente e una password per poter effettuare l'ordine.

Dopo aver effettuato l'ordine, il cliente può lasciare una recensione, con una valutazione da 1 a 5 e un commento facoltativo.

L'ordine è anch'esso identificato da un ID, e prevede solo l'ordinazione di un'auto. Si può riferire a un possibile acquisto, noleggio o leasing.

L'acquisto si caratterizza dal pagamento per intero dell'auto, con la possibilità di effettuare un finanziamento.

Il noleggio prevede il prestito dell'auto per un periodo limitato.

Il leasing, invece, prevede l'affitto dell'auto con possibilità di riscatto.

Ogni sede è identificata da un nome, e ogni auto può essere disponibile in solo una sede.

In ogni sede lavorano dei dipendenti (identificati dal codice fiscale), che si suddividono in manager, venditori e impiegati.

Ogni sede ha uno e un solo manager, e degli impiegati che si occupano di diverse mansioni.

Riguardo i venditori, inoltre, si memorizzano le auto vendute da ognuno.

Nel conto si memorizzano tutte le transazioni in entrata (ordini dei clienti) e in uscita (stipendi e acquisti auto), identificate da un ID univoco.

2.2. Glossario dei Termini

Termine	Descrizione	Collegamento
Cliente	Persone che effettuano ordini alla concessionaria	Collegata ad Ordine, Recensione
Ordine	Clienti che effettuano ordini per acquistare, noleggiare o prendere un'auto in leasing	Collegata a Cliente, Recensione, Auto, Pagamenti, Acquisto, Noleggio, Leasing
Recensione	Valutazione del cliente sulla qualità dell'ordine effettuato	Collegata a Cliente, Ordine
Acquisto	Tipologia di ordine che permette l'acquisto totale dell'auto.	Collegata a Ordine
Noleggio	Tipologia di ordine che permette l'utilizzo dell'auto per un periodo prestabilito	Collegata a Ordine
Leasing	Tipologia di ordine che prevede l'affitto dell'auto con un pagamento su più rate, con una possibilità di riscattare l'auto a fine contratto	Collegata a Ordine
Sede	Lista delle varie sedi della concessionaria	Collegata a Auto e Dipendente

Auto	Lista delle auto disponibili nella concessionaria	Collegata a Ordine, Sede e Acquisto_auto
Dipendente	Personale che lavora nella concessionaria	Collegato a Sede, Manager, Venditore, Impiegato e Conto
Manager	Dipendente della concessionaria che ha in gestione una delle sedi	Collegato a Dipendente
Venditore	Dipendente della concessionaria che si occupa delle vendite	Collegato a Dipendente
Impiegato	Dipendente della concessionaria che si occupa di altre mansioni	Collegato a Dipendente
Conto	Conto bancario della concessionaria: qui sono presenti tutte le transazioni in entrata e in uscita	Collegato a Ordine, Dipendente e Acquisto_auto
Acquisto_auto	Lista delle auto acquistate a prezzo di costo dalla concessionaria	Collegato a Conto e Auto

3. Progettazione concettuale

3.1 Lista delle entità

Il database rappresenta le seguenti classi. L'identificatore delle entità viene sottolineato e viene sott'inteso che sia *not null*.

- **Cliente:** rappresenta le persone che effettuano gli ordini
 - ID: serial primary key
 - Utente: varchar(20) not null
 - Password: varchar(20) not null
 - Nome: varchar(20) not null
 - Cognome: varchar(20) not null
- **Sede :** rappresenta le sedi della concessionaria
 - Nome: varchar(50) primary key
 - Stato: char(2) not null
 - Città: varchar(20) not null
 - Provincia: varchar(15) not null
 - Via: varchar(20) not null
 - N_civico: varchar(4) not null
- **Dipendente:** rappresenta i dipendenti della concessionaria
 - CF: varchar(16) primary key
 - Nome: varchar(20) not null
 - Cognome: varchar(20) not null
 - Sede: varchar(50) not null
 - Stipendio: decimal(6,2) not null
- **Manager:** dipendente che dirige una sede
 - CF: varchar(16) primary key
 - Inizio_dirigenza: date not null
 - Fine_dirigenza: date

- **Venditore:** dipendente che si occupa del settore vendite
 - CF: varchar(16) primary key
 - Auto_vendute: int
- **Impiegato:** dipendente che si occupa di altre mansioni
 - CF: varchar(16) primary key
 - Mansione: varchar(20) not null
- **Auto:** lista di auto presenti in concessionaria
 - ID: serial primary key
 - Marca: varchar(20) not null
 - Modello: varchar(20) not null
 - Colore: varchar(20) not null
 - Allestimento: varchar(20) not null
 - Sede: varchar(50) not null
 - Prezzo: decimal(8,2) not null
- **Ordine:** rappresenta gli ordini che vengono fatti dai clienti
 - ID: serial primary key
 - Auto: serial not null
 - Cliente: serial not null
- **Acquisto_auto:** auto acquistate dalla concessionaria
 - Auto: serial primary key
 - Prezzo_listino: decimal(8,2) not null
- **Conto:** conto della concessionaria che registra le transazioni
 - ID: serial primary key
 - Ordine: int
 - Stipendio: varchar(16)
 - Acquisto_auto: int
 - Operazione: boolean (T=entrata, F=uscita) not null
 - Data: timestamp not null
 - Importo: decimal(8,2) not null
- **Acquisto:** tipologia di ordine che permette l'acquisto dell'auto
 - Ordine: serial primary key
 - Prezzo: decimal(8,2) not null
 - Tasso: decimal(3,2)
 - Rata_mensile: decimal(5,2)
- **Noleggio:** tipologia di ordine che prevede il prestito dell'auto per un periodo limitato
 - Ordine: serial primary key
 - Data_inizio: date not null
 - Data_fine: date not null
 - Prezzo: decimal (6,2) not null
- **Leasing:** tipologia di ordine che prevede l'affitto dell'auto con possibilità di riscatto
 - Ordine: serial primary key
 - Anticipo: decimal(6,2) not null
 - Rata: decimal(6,2) not null
 - Riscatto: decimal (6,2)
 - Inizio_contratto: date not null
 - Fine_contratto: date not null
- **Recensione:** valutazione del cliente sulla qualità dell'ordine
 - ID Ordine: serial primary key
 - ID Cliente: serial primary key
 - Valutazione: int not null
 - Commento: varchar(100)

3.2 Tabella delle relazioni

Relazione	Entità coinvolte	Descrizione	Attributi
effettua	Cliente (0,N) Ordine (1,1)	Un cliente può effettuare 0 o più ordini. Un ordine viene effettuato da un solo cliente.	nessun attributo
scrive	Cliente (0,N) Recensione (1,1)	Un cliente può scrivere 0 o più recensioni. Una recensione può essere scritta da un solo cliente.	nessun attributo
relativo	Recensione (1,1) Ordine (0,1)	Una recensione è relativa a un solo ordine. Un ordine è relativo a 1 o 0 recensioni.	nessun attributo
comprende	Ordine (1,1) Auto (0,1)	Un ordine comprende una sola auto. Un' auto può essere o meno compresa in un ordine.	nessun attributo
disponibilità	Auto (0,1) Sede (1,N)	Un' auto può essere o meno disponibile in una sede. In una sede sono disponibili una o più auto.	nessun attributo
lavorano	Sede (1,N) Dipendente (1,1)	In una sede lavorano 1 o più dipendenti. Un dipendente lavora in una sola sede.	nessun attributo
pagato	Dipendenti (1,1) Conto (0,N)	Un dipendente è pagato tramite un solo conto. Un conto paga 0 o più dipendenti.	nessun attributo
compra	Conto (0,N) Auto (0,1)	Tramite il conto si possono comprare 0 o più auto. Un' auto può o meno essere comprata tramite il conto.	Prezzo_listino
pagamento	Ordine (1,1) Conto (0,N)	Un ordine viene pagato al conto. Al conto vengono pagati 0 o più ordini.	nessun attributo

3.3 Generalizzazioni

Ordine ← Acquisto
 Ordine ← Noleggio
 Ordine ← Leasing

Un ordine può essere un acquisto di un'auto, un noleggio per un breve periodo o un contratto di leasing: per questo viene suddiviso in tre sottocategorie con una generalizzazione totale.

Acquisto ⇌ Finanziamento

L'acquisto di un'auto può essere effettuato tramite un finanziamento. Per questo viene aggiunta una generalizzazione parziale.

Dipendente ← Manager
 Dipendente ← Venditore
 Dipendente ← Impiegato

Un dipendente può essere un manager di una sede, un venditore di auto o un impiegato: per questo viene suddiviso in tre sottocategorie con una generalizzazione totale.

[illegible]

4. Progettazione logica

4.1 Ristrutturazione dello schema

4.1.1 Analisi delle ridondanze

Il campo mensilità dell'entità leasing crea una ridondanza in quanto questo dato può essere ricavato sottraendo la data finale del contratto con la data iniziale, e dividendo tutto per 30 (dato che la semplice sottrazione mi avrebbe dato il numero dei giorni di differenza).

Anche il campo prezzo_totale dell'entità leasing è ridondante: infatti, può essere calcolato sommando l'anticipo alla moltiplicazione della rata per il numero di mesi trovato in precedenza.

Query per ottenere il numero di mesi di contratto e il totale del prezzo del contratto leasing:

```
select o.ID, o.Auto, l.inizio_contratto, l.fine_contratto, l.anticipo, l.rata,
((l.fine_contratto-l.inizio_contratto)/30) as durata_mesi,
(l.anticipo + (l.rata*((l.fine_contratto-l.inizio_contratto)/30))) AS costo_totale
from Ordine o join Leasing l on o.ID=l.Ordine
```

L'output ottenuto è il seguente:

Data Output		Explain	Messages	Notifications				
	id integer	 auto integer	 inizio_contratto date	 fine_contratto date	 anticipo numeric (6,2)	 rata numeric (6,2)	 durata_mesi integer	 costo_totale numeric
1	3	2	2021-02-18	2022-02-18	4500.00	1023.00	12	16776.00
2	4	6	2021-03-01	2022-03-01	7000.00	1200.00	12	21400.00

4.1.2 Eliminazione delle generalizzazioni

Generalizzazione	Risoluzione
Ordine ← Acquisto, Noleggio, Leasing	Si sceglie di creare tre tabelle separate in quanto sono entità distinte (Acquisto, Noleggio e Leasing), in quanto sono tre modalità diverse di ordine e presentano attributi diversi.
Acquisto ⇌ Finanziamento	L'entità finanziamento viene eliminata, in quanto il finanziamento potrebbe non essere effettuato. Quindi vengono spostati gli attributi nell'entità padre, permettendo valori nulli.
Dipendente ← Manager, Venditore, Impiegato	Si sceglie di creare tre tabelle separate in quanto sono entità distinte (Manager, Venditore, Impiegato), in quanto sono tre tipi diversi di dipendente e presentano attributi diversi.

In entrambe le generalizzazioni totali non è possibile eliminare l'entità padre e duplicare gli attributi, in quanto molti attributi dell'entità padre sono chiavi esterne di altre entità.

Si sarebbero anche dovute triplicare tutte le relazioni con le entità vicine, e sarebbe risultato tutto poco pulito.

In particolare, riguardo le entità Ordine - Recensione - Cliente, non si sarebbe potuto eliminare l'entità Ordine in quanto la recensione si deve riferire a un ordine. Inoltre, un cliente effettua un ordine, a prescindere se sia un acquisto, noleggio o Leasing.

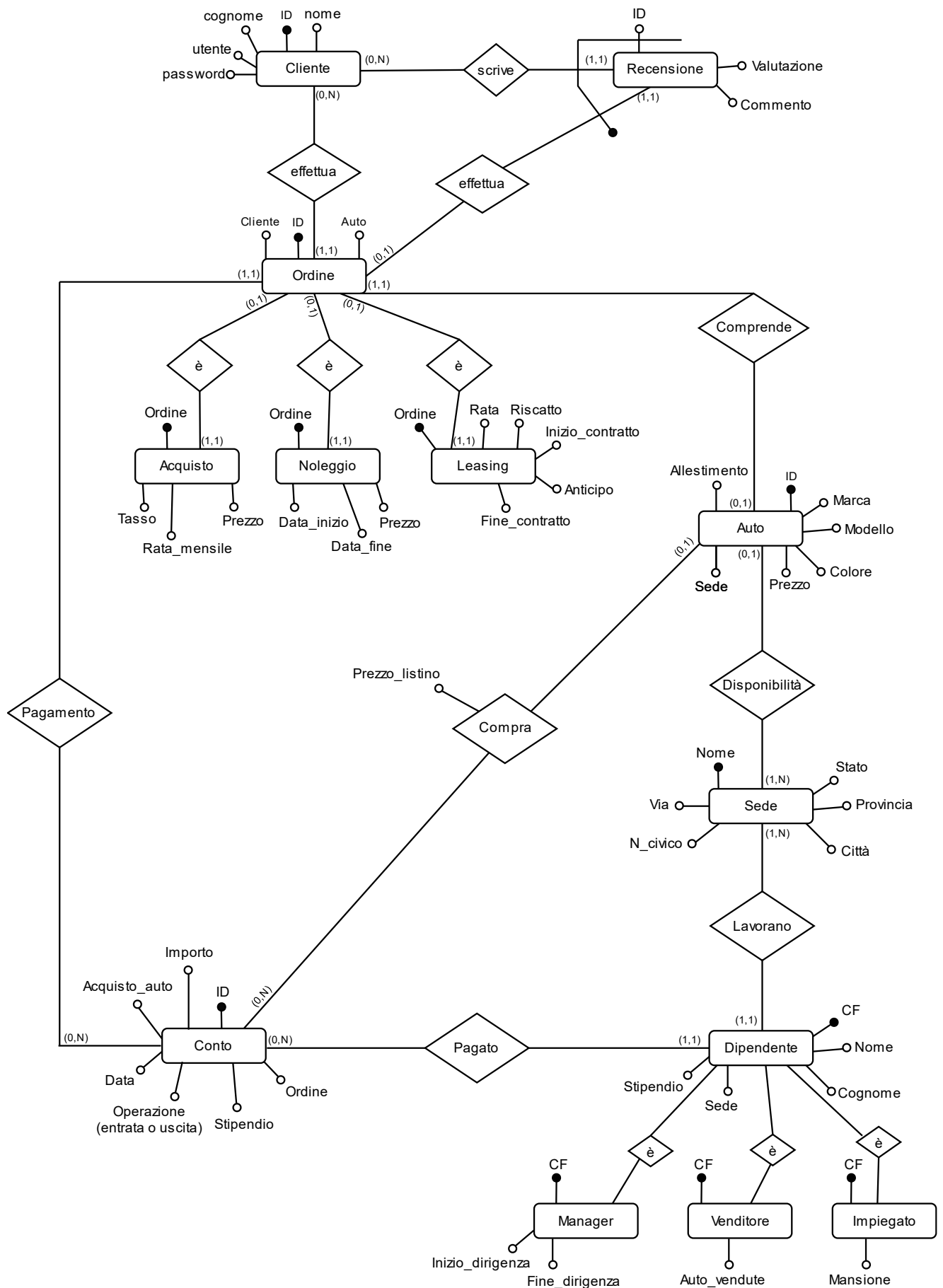
4.1.3 Scelta degli identificatori primari

Una chiave primaria non scontata è la chiave primaria della tabella recensione. Si è optato per la combinazione del campo ID, del campo ID_Ordine e del campo ID_Cliente, per evitare di avere ripetizioni di recensioni relativi a stessi ordini e clienti.

Se avessimo scelto di tenere come unica chiave primaria ID, avremmo potuto avere più recensioni dello stesso ordine fatto da un cliente (dove invece per ogni ordine si può scrivere una sola recensione). Inoltre, come ipotesi peggiore, avremmo potuto avere più recensioni dello stesso ordine fatte da clienti diversi (cosa non ammissibile in quanto un ordine può essere effettuato da un solo cliente).

Oltretutto, anche nella tabella cliente avremmo potuto scegliere come chiave primaria l'unione dei campi ID e utente: questo perché l'ordinazione viene effettuata online, e quindi oltre al fatto che l'ID deve essere univoco, anche l'utente non deve essere ambiguo. Però per evitare l'ambiguità dell'attributo utente, abbiamo optato per definirlo unico (UNIQUE).

4.1.4 Diagramma schema ristrutturato



4.2 Schema relazionale

cliente (ID, utente, password, nome, cognome)

sede (Nome, stato, provincia, via, n_civico)

dipendente (CF, nome, cognome, sede, stipendio)

manager (CF, inizio_dirigenza, fine_dirigenza)

venditore (CF, auto_vendure)

impiegato (CF, mansione)

auto (ID, marca, modello, colore, allestimento, sede, prezzo)

ordine (ID, auto, cliente)

acquisto_auto (auto, prezzo_listino)

conto (ID, ordine, stipendio, acquisto_auto, operazione, data, importo)

acquisto (ordine, prezzo, tasso, rata_mensile)

noleggio (ordine, data_inizio, data_fine, prezzo)

leasing (ordine, anticipo, rata, riscatto, inizio_contratto, fine_contratto)

recensione (ID, ID_ordine, ID_cliente, valutazione, commento)

sede.dipendente → sede.nome

manager.CF → dipendente.CF

venditore.CF → dipendente.CF

impiegato.CF → dipendente.CF

auto.sede → sede.nome

ordine.auto → auto.ID

ordine.cliente → cliente.ID

acquisto_auto.auto → auto.ID

conto.ordine → ordine.ID

conto.stipendio → dipendente.CF

conto.acquisto_auto → acquisto_auto.auto

acquisto.ordine → ordine.ID

noleggio.ordine → ordine.ID

leasing.ordine → ordine.ID

recensione.ID_cliente → cliente.ID


recensione.ID_ordine → ordine.id

5. Query e indice

5.1 Query

1. Selezionare gli ID delle operazioni in entrata del conto, l'ID dell'ordine di riferimento, la marca e il modello dell'auto acquistata dove l'importo è superiore a 25000 euro, ordinati per ordine.

```
SELECT c.ID AS ID_conto, o.ID AS ID_ordine, a.Marca, a.Modello
FROM conto c, ordine o JOIN auto a on o.auto=a.ID
WHERE c.ordine=o.ID and c.operazione=TRUE
      and c.importo>25000
ORDER BY o.ID
```

	Data Output	Explain	Messages	Notifications
	 id_conto integer 🔒	id_ordine integer 🔒	marca character varying (20) 🔒	modello character varying (20) 🔒
1	5	1	Volkswagen	Golf
2	1	3	Mercedes	A150
3	6	6	Nissan	Juke

2. Per ogni sede, selezionare il totale dei dipendenti che non sono manager, e, tra questi, lo stipendio maggiore percepito con il rispettivo nome e cognome del dipendente.

```
drop view if exists impiegati_per_sede;
```

```
CREATE VIEW impiegati_per_sede (sede, dipendenti, salario) as
SELECT s.nome, COUNT(d.CF), MAX(d.stipendio)
FROM sede s, dipendente d
WHERE s.nome=d.sede and d.CF NOT IN (SELECT CF
                                     FROM manager)
GROUP BY s.nome;
```

```
SELECT i.sede, i.dipendenti, i.salario, d.nome, d.cognome
FROM impiegati_per_sede i JOIN dipendente d ON i.sede=d.sede
WHERE i.salario=d.stipendio
GROUP BY i.sede, i.dipendenti, i.salario, d.nome, d.cognome
```

Data Output						Explain	Messages	Notifications
	sede character varying (50)	dipendenti bigint	salario numeric	nome character varying (20)	cognome character varying (20)			
1	Auto Lombardia	4	2436.19	Marco	Cecchi			
2	Car Center	4	2964.34	Sara	Castelli			
3	Dolomiti	4	2654.39	Davide	Filippin			
4	Smeralda	4	2631.02	Claudia	Pasquale			

3. Estrarre nome e cognome dei clienti (ordinati per cognome) che hanno effettuato un ordine d'acquisto su cui hanno scritto una recensione. Estrarre anche l'ID e la valutazione della recensione

```
drop view if exists clienti_acquisto;

CREATE VIEW clienti_acquisto(cognome, nome, ordine) AS
SELECT c.cognome, c.nome, o.id
FROM cliente c, ordine o
WHERE c.id=o.cliente

EXCEPT
SELECT c.cognome, c.nome, o.id
FROM cliente c, ordine o, noleggio, leasing
WHERE c.id=o.cliente AND o.id=noleggio.ordine OR o.id=leasing.ordine

ORDER BY cognome;

SELECT ca.cognome, ca.nome, a.prezzo, r.id as id_recensione, r.valutazione
FROM clienti_acquisto ca, acquisto a, recensione r
WHERE ca.ordine=r.ID_ordine AND a.ordine=ca.ordine
```

Data Output						Explain	Messages	Notifications
	<div><div></div><div>cognome</div><div>character varying (20)</div></div>	<div><div></div><div>nome</div><div>character varying (20)</div></div>	<div><div></div><div>prezzo</div><div>numeric (8,2)</div></div>	<div><div></div><div>id_recensione</div><div>integer</div></div>	<div><div></div><div>valutazione</div><div>integer</div></div>			
1	Piruli	Alessia	18345.00	4	3			
2	Tassi	Alfredo	25647.00	1	4			

4. Estrarre le sedi con il prezzo medio delle auto superiore a 26000 euro. Mostrare il numero di auto disponibili per ogni sede, e il prezzo medio.

```
drop view if exists auto_per_sede;
```

```
CREATE VIEW auto_per_sede (sede, tot_auto) AS  
SELECT a.sede, count(*)  
FROM auto a  
GROUP BY a.sede;
```

```
SELECT aps.sede, aps.tot_auto, ROUND(AVG(a.prezzo), 2) AS prezzo_medio  
FROM auto_per_sede aps, auto a  
WHERE a.sede=aps.sede  
GROUP BY aps.sede, aps.tot_auto  
HAVING AVG(a.prezzo)>26000
```

	Data Output	Explain	Messages	Notifications
	sede character varying (50)	tot_auto bigint	prezzo_medio numeric	
1	Dolomiti	3	27399.67	
2	Smeralda	3	26676.67	

5. Selezionare per ogni sede la somma di tutti gli stipendi dei dipendenti e il nome e cognome del manager che la dirige.

```
drop view if exists stipendi_per_sede;
```

```
CREATE VIEW stipendi_per_sede(sede, somma_stipendi) AS  
SELECT s.nome, sum(d.stipendio)  
FROM sede s, dipendente d  
WHERE s.nome=d.sede  
GROUP BY s.nome;
```

```
SELECT ss.sede, (d.cognome, d.nome) AS manager, ss.somma_stipendi  
FROM stipendi_per_sede ss, dipendente d, manager m  
WHERE d.CF=m.cf AND d.sede=ss.sede
```

	Data Output	Explain	Messages	Notifications
	sede character varying (50)	manager record	somma_stipendi numeric	
1	Car Center	(Evans,Chris)	14533.62	
2	Smeralda	(Rini,Silvio)	13108.35	
3	Auto Lombardia	(Biagio,Anna)	11504.99	
4	Dolomiti	("De Lorenzi",Alberto)	12623.72	

6. Estrarre dal conto l'ultimo movimento che è stato effettuato in ordine di tempo. Visualizzare il numero del movimento (ID del conto), l'eventuale ordine, codice fiscale del dipendente in caso di stipendio o l'acquisto dell'auto, e il prezzo. Riguardo al prezzo, se l'operazione è di uscita, visualizzare il segno meno per indicarlo.






```
drop view if exists segno_importo;

CREATE VIEW segno_importo (movimento, importo) AS
SELECT c.id, (0-c.importo) AS importo
FROM conto c
WHERE c.operazione=FALSE

UNION

SELECT c.id, c.importo
FROM conto c
where c.operazione=TRUE;

SELECT c.ID as Movimento, c.ordine, c.stipendio AS stipendio_dipendente,
       c.acquisto_auto, s.importo
FROM Conto c, segno_importo s
WHERE c.id=s.movimento AND c.data=(SELECT MAX(c.data)
                                     FROM conto c)
```

Data Output		Explain	Messages	Notifications	
	movimento integer	 ordine integer	 stipendio_dipendente character varying (16)	 acquisto_auto integer	 importo numeric
1	12	[null]	GDIARO96O08P025F	[null]	-2469.72

Nel caso si volesse scegliere di visualizzare un giorno specifico, si può utilizzare l'operatore di uguaglianza nel WHERE. Per il giorno corrente, invece, si può semplicemente utilizzare la dicitura *current_date* al posto dell'operatore di uguaglianza.

Per quanto riguarda il file .cpp, per il corretto funzionamento dello stesso, il Database deve essere creato e popolato attraverso lo script fornito. Dopo di che, bisogna assicurarsi che i “**#define**” siano coerenti con quanto impostato nella propria macchina. Questo perché per il corretto funzionamento del codice C++, c'è bisogno della libreria “**libpq-fe**” che abilita all'utilizzo di funzioni che permettono di interagire con il Database. Dopo aver fatto tutto ciò, si può compilare lo script ed eseguire il file eseguibile creato per interrogare il database con le query soprantanti.

5.2 Indici

Importante è una ricerca dettagliata dei vari movimenti nel conto corrente, in quanto in poco tempo si accumuleranno sempre di più: si crea un indice per la ricerca di tali movimenti, basata su numero dell'ordine, numero dell'auto acquistata oppure del codice fiscale del dipendente se si tratta rispettivamente di un ordine, di un acquisto di un'auto o di un salario. Inoltre si può ricercare anche per importo.

```
drop index if exists ricerca_movimento;  
create index ricerca_movimento on Conto(Ordine, Stipendio, Acquisto_auto, Importo);
```

Allo stesso modo anche la ricerca dei dipendenti potrà necessitare di un indice. Ogni dipendente potrà essere ricercato per nome, cognome o sede in cui lavora.

```
drop index if exists ricerca_dipendente;  
create index ricerca_dipendente on Dipendente(Nome, Cognome, Sede);
```

Con questi indici la ricerca su migliaia di movimenti o centinaia di dipendenti sarà più rapida e specifica.