Basi di Dati

Project: BestBooze

Del Moro Giacomo, Mat. 1123117, username: gdelmoro
Coletti Andrea, Mat. 1096089, username: acoletti

Il progetto è caricato sul Database 'gdelmoro-PR' di gdelmoro.

Indice

Abstract

Si vuole realizzare la base di dati del sito di e-commerce dell'azienda BestBooze che vende alcolici.

Un cliente è identificato univocamente da un Username, ha un nominativo, un indirizzo di spedizione, una mail, un recapito telefonico ed un eventuale numero di partita IVA.
I clienti possono acquistare i prodotti disponibili, che sono identificati da un codice, hanno un nome, un prezzo, una categoria, il loro anno di produzione e il nome del produttore.

La fase di acquisto è caratterizzata dall'inserimento di prodotti nel carrello, unico per ogni cliente, nel quale viene indicata la quantità desiderata per oggetto ed il costo dei prodotti inseriti.
Alla conferma dell'ordine viene emessa una fattura con intestatario il cliente effettuante l'ordine, con l'elenco dei prodotti e i rispettivi prezzi parziali, la data di emissione ed il costo totale, scontato in base al prezzo finale.

Viene poi scelta una modalità di spedizione a scelta tra tre opzioni: standard, che è gratuita ed ha un tempo di spedizione standard; express che ha un supplemento e ha tempi di spedizione brevi; daily che ha un supplemento più alto, ma garantisce la consegna entro un giorno lavorativo.

Gli ordini, una volta conclusi, vengono memorizzati in un archivio per tenere uno storico. Ciascuno viene salvato con il numero d'ordine, il cliente che lo ha effettuato, la data in cui è stato effettuato, la fattura associata, la modalità di spedizione scelta e il costo finale dell'ordine, somma del costo dei prodotti e delle spese di spedizione.

Analisi dei requisiti

Il progetto consiste nella realizzazione del sistema di gestione degli ordini del sito dell'azienda BestBooze che vende alcolici.

Il sito permette di effettuare ordini solo a clienti, che sono utenti registrati al sito.
Ogni cliente è identificato univocamente da un Username ed è caratterizzato dalle seguenti informazioni:

- Un indirizzo di spedizione;
- Un indirizzo email;
- Un recapito telefonico;
- Un eventuale partita IVA;

I prodotti acquistabili sono identificati da:

- Un codice, unico per ogni prodotto;
- Un nome;
- Una categoria che descriva il tipo di alcolico;
- Un anno di produzione;
- Il nome del produttore;
- Il prezzo del prodotto;

La parte fondamentale della base di dati consiste nel sistema di gestione degli ordini. Un ordine è rappresentato da:

- Un numero d'ordine;
- Il cliente ordinante;
- La data dell'ordine;

Gli ordini sono poi divisi in due categorie: ordini effettuati e ordini passati.

Un ordine in corso è collegato ad un Carrello che caratterizzato da:

- Un codice identificativo;
- L'elenco dei prodotti che un utente desidera acquistare;
- La quantità desiderata per ciascun prodotto;
- Il prezzo totale dei prodotti nel carrello;

Quando un ordine in corso viene confermato allora viene emessa una fattura, descritta da:

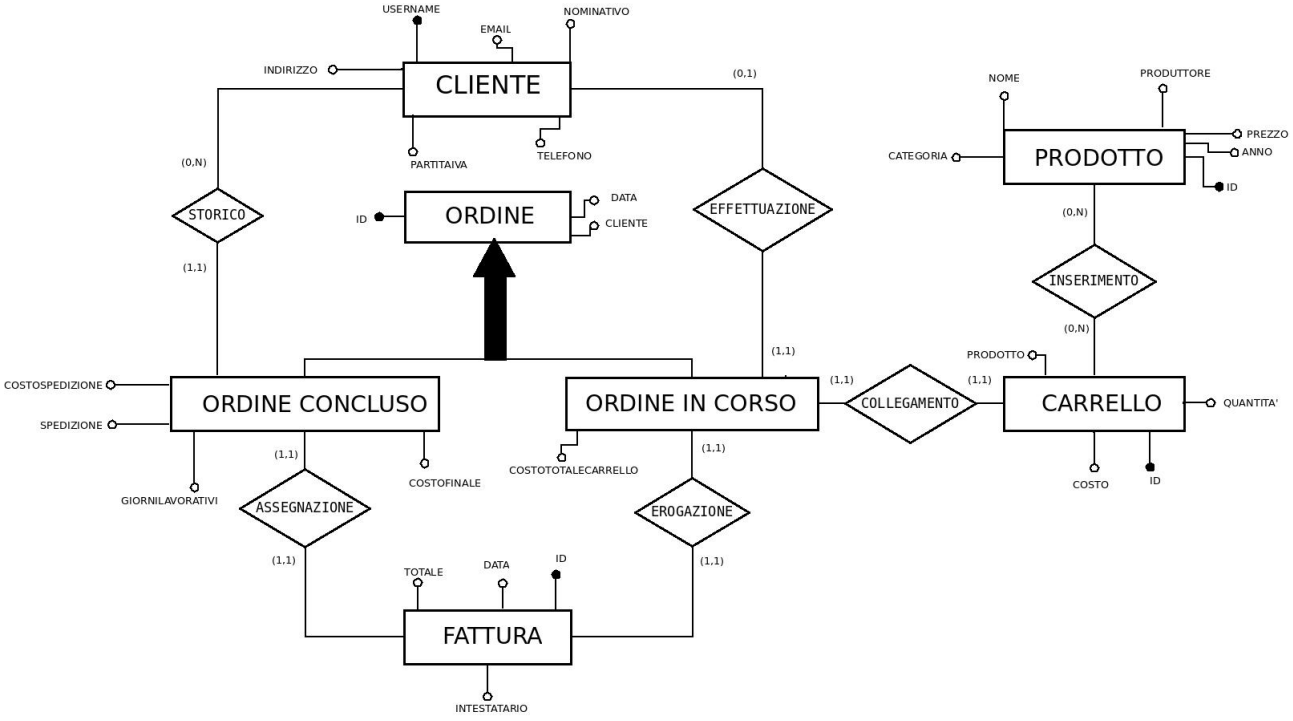
- Un numero di fattura;
- Il cliente intestatario;
- La data di emissione;
- Il prezzo della merce acquistata;

All'emissione della fattura, può venire applicato uno sconto, che varia in base al costo totale della merce acquistata. Dopodiché il cliente seleziona un metodo di spedizione per il suo ordine. Le modalità di

spedizione sono: Standard, gratuita e con consegna entro 7-10 giorni lavorativi; Express, con costo di 20€ e consegna entro 3-5 giorni lavorativi; Daily, che ha un supplemento di 50€ e garantisce la consegna entro un giorno lavorativo.
L'ordine viene quindi inserito negli ordini conclusi, archiviati con le seguenti caratteristiche aggiuntive rispetto all'ordine generico:

- La modalità di spedizione scelta;
- Il costo della spedizione;
- Il numero di giorni lavorativi previsti per la spedizione;
- Il costo totale, somma del totale della fattura e del costo di spedizione.

Schema Concettuale



Lista delle classi

Cliente: rappresenta il cliente registrato al sito che può effettuare ordini.
Attributi:

- Username: string;
- Nominativo: string;
- Indirizzo: string;
- Email: string;
- Telefono: string;
- Partita IVA: string;

Prodotto: rappresenta la merce acquistabile dai clienti.
Attributi:

- ID: int;
- Nome: string;
- Prezzo: float;
- Categoria: string;
- Anno di produzione: date;
- Produttore: string;

Carrello: è associato ad ogni ordine in corso e contiene quali e quanti prodotti vengono ordinati.
Attributi:

- ID: int;
- Prodotto: int;
- Quantità: int;
- Costo: float;

Fattura: contiene il costo totale dei prodotti in un ordine, l'intestatario e la data di fatturazione.
Attributi:

- ID: int;
- Intestatario: string;
- Totale: float;
- Data: date;

Ordine: rappresenta l'acquisto da parte di un cliente dei prodotti scelti nella quantità desiderata.

- Attributi:
  - ID: int;
  - Cliente: string;
  - Data: date;
- Sottoclassi:
  1) **Ordine in corso**: è l'ordine in esecuzione da parte di un cliente, mentre sta inserendo i prodotti nel carrello.
     Attributi:
     - Costo totale prodotti nel carrello: float;
  2) **Ordine concluso**: rappresenta tutti gli ordini conclusi, con la modalità di spedizione associata ed il costo finale.
     Attributi:
     - Spedizione: string;
     - Costo spedizione: int;
     - Giorni lavorativi: string;
     - Costo finale: float;

Lista delle relazioni

- Ordine in corso – Cliente: Esecuzione
  - Un cliente può eseguire al più un ordine alla volta;
  - Un ordine può essere effettuato da uno ed un solo cliente;

  Molteplicità: uno a uno (1:1)

- Ordine concluso – Cliente: Storico
  - Un cliente può avere nessuno o molti ordini conclusi;
  - Un ordine può essere effettuato da uno ed un solo cliente;

  Molteplicità: uno a molti (1:N)

- Ordine in corso – Fattura: Erogazione
  - Un ordine ha una ed una sola fattura associata;
  - Una fattura è associata solo ad un ordine;

  Molteplicità: uno a uno (1:1)

- Fattura – Ordine concluso: Assegnazione
  - Una fattura è assegnata ad un solo ordine;
  - Un ordine ha una ed una sola fattura associata;

  Molteplicità: uno a uno (1:1)

- Prodotto – Carrello: Inserimento
  - Un prodotto può essere inserito in nessuno o più carrelli;
  - Un carrello piò essere vuoto o avere più prodotti;

  Molteplicità: molti a molti (N:N)

- Carrello – Ordine in corso: Collegamento
  - Un carrello è collegato ad un solo ordine in corso;
  - Un ordine in corso può avere un solo carrello collegato.

  Molteplicità: uno a uno (1:1)

Schema Relazionale

Cliente (**Username**, Nominativo, Indirizzo, Email, Telefono, Partita IVA)

Prodotto (**ID**, Nome, Prezzo, Categoria, Anno, Produttore)

Carrello (**OrdineInCorso**, **Prodotto**, Quantità, Costo)

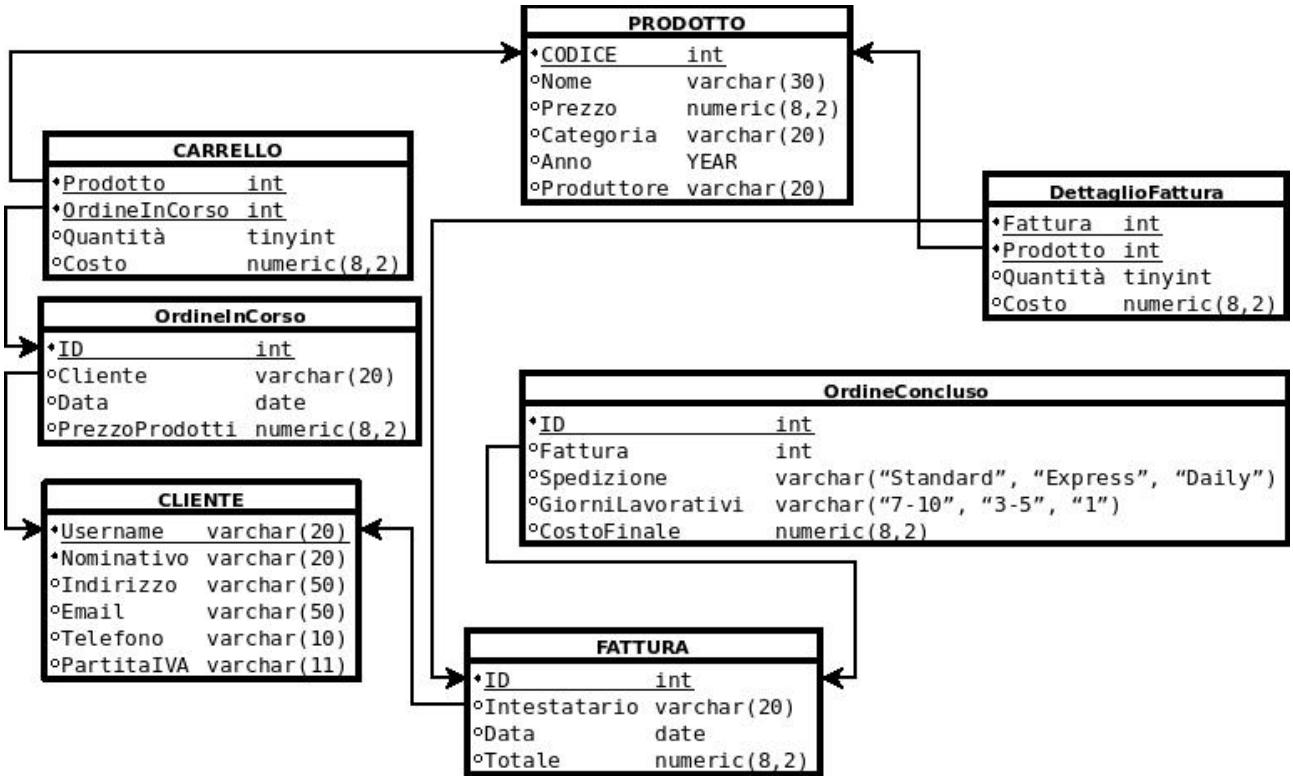
OrdineInCorso (**ID**, Cliente, Data, PrezzoProdotti)

Fattura (**ID**, Intestatario, Data, Totale)

DettaglioFattura (**Fattura**, **Prodotto**, Quantità, Costo)

OrdineConcluso (**ID**, Fattura, Spedizione, GiorniLavorativi, CostoFinale)

Schema Logico



Implementazione del Database

Creazione delle tabelle

```
DROP TABLE IF EXISTS Cliente;

CREATE TABLE Cliente (

    Username VARCHAR(20),
    Nominativo VARCHAR(50),
    Indirizzo VARCHAR(50),
    Email VARCHAR(50),
    Telefono VARCHAR(10),
    PartitaIVA VARCHAR(11) DEFAULT NULL,

    PRIMARY KEY (Username)
);
```

```
DROP TABLE IF EXISTS Prodotto;

CREATE TABLE Prodotto (

    Codice INT,
    Nome VARCHAR(30),
    Prezzo NUMERIC(8,2),
    Categoria VARCHAR(20),
    Anno YEAR DEFAULT NULL,
    Produttore VARCHAR(20),

    PRIMARY KEY (Codice)
);

DROP TABLE IF EXISTS Carrello;

CREATE TABLE Carrello (

    Prodotto INT,
    OrdineInCorso INT,
    Quantità SMALLINT,
    Costo NUMERIC(8,2),

    PRIMARY KEY (Prodotto, OrdineInCorso),
    FOREIGN KEY (Prodotto) REFERENCES Prodotto(Codice) ON DELETE CASCADE,
    FOREIGN KEY (OrdineInCorso) REFERENCES OrdineInCorso(ID) ON DELETE CASCADE
);

DROP TABLE IF EXISTS OrdineInCorso;

CREATE TABLE OrdineInCorso (

    ID INT,
    Cliente VARCHAR(20),
    Data DATE,
    PrezzoProdotti NUMERIC(8,2) DEFAULT 0,

    PRIMARY KEY (ID),
    FOREIGN KEY (Cliente) REFERENCES Cliente(Username) ON DELETE NO ACTION
);

DROP TABLE IF EXISTS Fattura;

CREATE TABLE Fattura (

    ID INT,
    Intestatario VARCHAR(20),
    Data DATE,
    Totale NUMERIC(8,2),

    PRIMARY KEY (ID),
    FOREIGN KEY (Intestatario) REFERENCES Cliente(Username) ON DELETE NO ACTION
);
```

```
DROP TABLE IF EXISTS DettaglioFattura;

CREATE TABLE DettaglioFattura (

    Fattura INT,
    Prodotto INT,
    Quantità SMALLINT,
    Costo NUMERIC(8,2),

    PRIMARY KEY (Fattura, Prodotto),
    FOREIGN KEY (Fattura) REFERENCES Fattura(ID) ON DELETE CASCADE,
    FOREIGN KEY (Prodotto) REFERENCES Prodotto(Codice) ON DELETE NO ACTION
);

DROP TABLE IF EXISTS OrdineConcluso;

CREATE TABLE OrdineConcluso (

    ID INT,
    Fattura INT,
    Spedizione ENUM ('Standard', 'Express', 'Daily'),
    GiorniLavorativi ENUM ('7-10', '3-5', '1'),
    CostoFinale NUMERIC(8,2),

    PRIMARY KEY (ID),
    FOREIGN KEY (Fattura) REFERENCES Fattura(ID) ON DELETE NO ACTION
);
```

Query, Procedure, Funzioni

Di seguito elenchiamo tutte le Query, Procedure e Funzioni utilizzate nel Database.
Mostriamo anche l'Output per tutte le Query e Procedure eseguite sul Database popolato con dati di esempio.

Funzione MPC: Funzione di utilità che mostra il prezzo medio dei prodotti di una Categoria passata come parametro di Input.

```
DELIMITER |

DROP FUNCTION IF EXISTS MPC |

CREATE FUNCTION MPC (Categoria VARCHAR(20)) RETURNS NUMERIC(8,2)

      BEGIN

            DECLARE val NUMERIC(8,2);
            SELECT AVG(P.Prezzo) INTO val
            FROM Prodotto P
            WHERE P.Categoria = Categoria;

            RETURN val;

      END |

DELIMITER ;
```

Funzione MPC: Funzione di utilità che mostra il prezzo massimo tra i prodotti di una Categoria passata come parametro di Input.

```
DELIMITER |

DROP FUNCTION IF EXISTS PMAXC |

CREATE FUNCTION PMAXC (Categoria VARCHAR(20)) RETURNS NUMERIC(8,2)

       BEGIN

              DECLARE val NUMERIC(8,2);
              SELECT MAX(P.Prezzo) INTO val
              FROM Prodotto P
              WHERE P.Categoria = Categoria;

              RETURN val;

       END |

DELIMITER ;
```

Funzione MPC: Funzione di utilità che mostra il prezzo minimo tra i prodotti di una Categoria passata come parametro di Input.

```
DELIMITER |

DROP FUNCTION IF EXISTS PMINC |

CREATE FUNCTION PMINC (Categoria VARCHAR(20)) RETURNS NUMERIC(8,2)

       BEGIN

              DECLARE val NUMERIC(8,2);
              SELECT MIN(P.Prezzo) INTO val
              FROM Prodotto P
              WHERE P.Categoria = Categoria;

              RETURN val;

       END |

DELIMITER ;
```

Query 1: Mostra le Email dei clienti che hanno speso in almeno un ordine un valore maggiore della media dei prezzi della categoria 'Birra' senza mostrare coloro che hanno speso meno del 5% del vino più costoso. Si mostri anche il totale speso per ogni ordine che soddisfa i requisiti.

```
SELECT C.Email, F.Totale
FROM Cliente C, Fattura F
WHERE F.Intestatario = C.Username
      AND F.Totale>MPC('Birra')
      AND C.Username NOT IN (SELECT C.username
                             FROM Cliente C, Fattura F
                             WHERE F.Intestatario = C.Username
                             AND F.Totale < (PMAXC('Vino')*0.05));
```

Output:

```
+-----------------------+-----------+
| Email                 | Totale    |
+-----------------------+-----------+
| discoteca@discostu.it |  45174.60 |
| maztek@maztek.com     | 162250.92 |
| mconti@mail.it        | 134615.70 |
| maztek@maztek.com     |  47232.00 |
| maztek@maztek.com     | 170775.00 |
+-----------------------+-----------+
```

Query 2: Si mostri il nominativo dei clienti che hanno concluso ordini nell'ultimo anno, con almeno un prodotto il cui prezzo è minore di 1000€, ad eccezione di vini. Si mostri anche il nome del prodotto. L'ordine deve avere inoltre la spedizione di tipo Express.

```
DROP VIEW IF EXISTS OrdiniAnno;

CREATE VIEW OrdiniAnno AS

       SELECT F.ID
       FROM Fattura F, OrdineConcluso OC
       WHERE OC.Fattura = F.ID
             AND F.Data >= (SELECT DATE_SUB(CURDATE(), INTERVAL 1 YEAR))
             AND OC.Spedizione = 'Express';

SELECT P.Nome AS Prodotto, C.Nominativo AS Cliente
FROM DettaglioFattura DF, Cliente C, Prodotto P, Fattura F, OrdiniAnno OA
WHERE DF.Fattura = OA.ID
      AND F.ID = OA.ID
      AND F.Intestatario = C.Username
      AND P.Codice = DF.Prodotto
      AND DF.Costo <= '1000.00'
      AND P.Categoria <> 'Vino'
ORDER BY Prodotto;
```

Output:

```
+---------------+---------------+
| Prodotto      | Cliente       |
+---------------+---------------+
| Bacardi       | Maztek S.p.a. |
| Dreher        | Maztek S.p.a. |
| Gin Fizz      | Maztek S.p.a. |
| Harp Lager    | Maztek S.p.a. |
| Jägermeister  | Maztek S.p.a. |
| Montenegro    | Maztek S.p.a. |
| Old Fashioned | Maztek S.p.a. |
+---------------+---------------+
```

Funzione che dato l'Username di un utente restituisce il costo medio degli ordini di quel cliente.

```
DELIMITER |

DROP FUNCTION IF EXISTS GiveAVG |

CREATE FUNCTION GiveAVG (usr VARCHAR(20)) RETURNS NUMERIC(8,2)

	BEGIN

		DECLARE Media NUMERIC(8,2);
		SELECT AVG(Totale) INTO Media
		FROM Fattura
		WHERE Intestatario = usr;

		RETURN Media;

	END |

DELIMITER ;
```

Funzione che calcola il numero di ordini effettuato da un cliente dato il suo username.

```
DELIMITER |

DROP FUNCTION IF EXISTS NumOrdini |

CREATE FUNCTION NumOrdini (usr VARCHAR(20)) RETURNS TINYINT

	BEGIN

		DECLARE NumOrdini NUMERIC(8,2);
		SELECT COUNT(*) INTO NumOrdini
		FROM Fattura
		WHERE Intestatario = usr;

		RETURN NumOrdini;

	END |

DELIMITER ;
```

Funzione che, a seconda del parametro di ingresso, restituisce il nome del cliente con più ordini o il nome di quello con meno ordini. A parità di numero di ordini si sceglie il cliente con costo medio più alto o più basso a seconda che si sia scelto rispettivamente il cliente con più ordini o quello con meno.

```
DROP VIEW IF EXISTS UtentiDistinti;

CREATE VIEW UtentiDistinti AS

	SELECT Intestatario
	FROM Fattura
	GROUP BY Intestatario;

DROP VIEW IF EXISTS ResocontoOrdini;
```

```
CREATE VIEW ResocontoOrdini AS

	SELECT C.Nominativo AS Cliente, NumOrdini(UD.Intestatario) AS NumeroOrdini,
		GiveAVG(UD.Intestatario) AS MediaOrdini
	FROM Cliente C, UtentiDistinti UD
	WHERE C.Username = UD.Intestatario;

DELIMITER |

DROP FUNCTION IF EXISTS MaxMinOrdini |

CREATE FUNCTION MaxMinOrdini (MinMax ENUM('min', 'max')) RETURNS VARCHAR(20)

	BEGIN

		DECLARE nome VARCHAR(20);

		IF MinMax = 'max'
		THEN
			SELECT Cliente INTO nome
			FROM ResocontoOrdini
			WHERE MediaOrdini = (SELECT MAX(RO.MediaOrdini)
					     FROM ResocontoOrdini RO
					     WHERE NumeroOrdini =
						   (SELECT MAX(RO.NumeroOrdini)
						    FROM ResocontoOrdini RO));
		ELSE
			SELECT Cliente INTO nome
			FROM ResocontoOrdini
			WHERE MediaOrdini = (SELECT MIN(RO.MediaOrdini)
					     FROM ResocontoOrdini RO
					     WHERE NumeroOrdini =
						   (SELECT MIN(RO.NumeroOrdini)
						    FROM ResocontoOrdini RO));
		END IF;

		RETURN nome;

	END |

DELIMITER ;
```

Query 3: Seleziona il numero medio di ordini effettuati per cliente, il nome del cliente che ha effettuato più ordini e il cliente che ha effettuato meno ordini. A parità di numero di ordini, selezionare il cliente che ha la spesa media più alta per il cliente che ha effettuato più ordini, la spesa minima per il cliente che ha effettuato meno ordini.

```
SELECT AVG(RO.NumeroOrdini) AS NumeroMedioOrdini, MaxMinOrdini('max') AS ClienteMAX,
	MaxMinOrdini('min') AS ClienteMIN
FROM ResocontoOrdini RO;
```

Output:

```
+------------------+--------------+--------------------+
| NumeroMedioOrdini | ClienteMAX    | ClienteMIN         |
+------------------+--------------+--------------------+
|           1.4286 | Maztek S.p.a. | Genoveffa Marquina |
+------------------+--------------+--------------------+
```

Query 4: Mostra l'email dei clienti residenti a Padova che hanno effettuato almeno un ordine il cui costo sia maggiore o uguale del prezzo medio degli ordini dei clienti di Milano.

```
SELECT C.Email
FROM Cliente C, Fattura F
WHERE C.Username = F.Intestatario
      AND C.Indirizzo LIKE '%, Padova'
      AND NumOrdini(Username) >= 1
      AND F.Totale >= (SELECT AVG(Totale)
                       FROM Cliente C, Fattura F
                       WHERE C.Username = F.Intestatario
                       AND C.Indirizzo like '%, Milano')
GROUP BY C.Email;
```

Output:

```
+------------------+
| Email            |
+------------------+
| gdelmoro@mail.it |
| mconti@mail.it   |
+------------------+
```

Procedura 1: Mostra per ogni categoria di Prodotto il guadagno totale derivato dalle vendite effettuate di quella categoria e la percentuale rispetto al guadagno totale dell'azienda. Si ignorino gli sconti applicati a fine ordine.

```
DELIMITER |

DROP PROCEDURE IF EXISTS PercentualeGuadagno |

CREATE PROCEDURE PercentualeGuadagno (tot NUMERIC(8,2))

       BEGIN
              DECLARE Done INT DEFAULT 0;
              DECLARE cat VARCHAR(20);
              DECLARE guad NUMERIC(8,2);

              DECLARE cat_cursor CURSOR FOR
                     SELECT P.Categoria
                     FROM Prodotto P
                     GROUP BY P.Categoria;
              DECLARE CONTINUE HANDLER FOR NOT FOUND SET Done = 1;

              DROP TEMPORARY TABLE IF EXISTS Aux;
              CREATE TEMPORARY TABLE Aux (Categoria VARCHAR(20),
                                          GuadagnoCategoria NUMERIC(8,2),
                                          GuadagnoInPercentuale NUMERIC(8,2));

              OPEN cat_cursor;

              REPEAT
                     FETCH cat_cursor INTO cat;
                     SELECT SUM(DF.Costo) AS GuadagnoCategoria INTO guad
                     FROM DettaglioFattura DF, Prodotto P
                     WHERE P.Categoria = cat
                            AND P.Codice = DF.Prodotto;
```

```
                IF NOT Done THEN
                      IF (SELECT P.Categoria
                          FROM DettaglioFattura DF, Prodotto P
                          WHERE P.Categoria = cat
                                AND P.Codice = DF.Prodotto
                          GROUP BY P.Categoria) IS NOT NULL
                      THEN
                            INSERT INTO Aux (Categoria, GuadagnoCategoria,
                                             GuadagnoInPercentuale)
                                   SELECT P.Categoria, SUM(DF.Costo),
                                          (100 * guad / tot)
                                   FROM DettaglioFattura DF, Prodotto P
                                   WHERE P.Categoria = cat
                                         AND P.Codice = DF.Prodotto;
                      END IF;
                END IF;
          UNTIL Done END REPEAT;
      CLOSE cat_cursor;

          SELECT * FROM Aux;
          DROP TEMPORARY TABLE Aux;

      END |

DELIMITER ;

CALL PercentualeGuadagno((SELECT SUM(Costo) FROM DettaglioFattura));
```

Output:

```
+---------------+-------------------+-----------------------+
| Categoria     | GuadagnoCategoria | GuadagnoInPercentuale |
+---------------+-------------------+-----------------------+
| Birra         |           3836.00 |                  0.48 |
| Cocktail      |           8559.80 |                  1.07 |
| Distillato    |           1332.00 |                  0.17 |
| Liquori       |           1597.80 |                  0.20 |
| Superalcolici |           3591.48 |                  0.45 |
| Vino          |         778574.50 |                 97.63 |
+---------------+-------------------+-----------------------+
```

Procedura 2: Mostra il nome del prodotto che è stato acquistato di più in un anno inserito in Input e il numero di clienti con e senza partita IVA che hanno acquistato quel prodotto.

```
DELIMITER |

DROP PROCEDURE IF EXISTS PiùVenduto |

CREATE PROCEDURE PiùVenduto (Anno YEAR)

      BEGIN

          DECLARE prod INT;
          DECLARE pIVA INT;
          DECLARE nonpIVA INT;

          DROP TABLE IF EXISTS VenditeProdotto;

          CREATE TABLE VenditeProdotto (
                Prodotto INT,
                NumeroOrdini INT,
                QuantitàTotalE INT
          );
```

```
            INSERT INTO VenditeProdotto (SELECT P.Codice, COUNT(DF.Quantità),
                                                SUM(DF.Quantità)
                                         FROM DettaglioFattura DF JOIN Prodotto P ON
                                              (DF.Prodotto = P.Codice) JOIN Fattura F ON
                                              (F.ID = DF.Fattura)
                                         WHERE YEAR(F.Data) = Anno
                                         GROUP BY DF.Prodotto);

            SELECT Prodotto INTO prod
            FROM VenditeProdotto
            WHERE QuantitàTotale = (SELECT MAX(QuantitàTotale)
                                    FROM VenditeProdotto);

            SELECT COUNT(*) INTO pIVA
            FROM Fattura F, DettaglioFattura DF, Cliente C
            WHERE DF.Prodotto = prod
                  AND F.ID = DF.Fattura
                  AND YEAR(F.Data) = Anno
                  AND C.Username = F.Intestatario
                  AND C.PartitaIva IS NOT NULL;

            SELECT COUNT(*) INTO nonpIVA
            FROM Fattura F, DettaglioFattura DF, Cliente C
            WHERE DF.Prodotto = prod
                  AND F.ID = DF.Fattura
                  AND YEAR(F.Data) = Anno
                  AND C.Username = F.Intestatario
                  AND C.PartitaIva IS NULL;


            SELECT P.Nome AS PiùVenduto, pIVA AS N_pIVA, nonpIVA AS N_non_pIVA
            FROM Prodotto P
            WHERE P.Codice = prod;

            DROP TABLE VenditeProdotto;

      END |

DELIMITER ;

CALL PiùVenduto (2015);
```

Output:

```
+-------------+--------+------------+
| PiùVenduto  | N_pIVA | N_non_pIVA |
+-------------+--------+------------+
| Harp Lager  |      2 |          1 |
+-------------+--------+------------+
```

Trigger

Trigger 1: Dopo l'inserimento di una entry in Fattura si applica uno sconto al prezzo totale se l'ordine rientra nei requisiti. Se ha speso meno di 200€ lo sconto non si applica. Lo sconto aumenta del 1% per ogni 200€ fino ad un massimo del 10%.

```
DELIMITER |

DROP TRIGGER IF EXISTS Sconto |

CREATE TRIGGER Sconto

BEFORE INSERT ON Fattura
FOR EACH ROW

	BEGIN

		DECLARE sc INT;

		SET sc = NEW.Totale DIV 200;

		IF sc > 10
			THEN SET sc = 10;
		END IF;

		IF NEW.Totale > 200
			THEN SET NEW.Totale = NEW.Totale - (NEW.Totale * sc / 100);
		END IF;

	END |

DELIMITER ;
```

Trigger 2: All'aggiornamento della quantità di un prodotto nel carello, e quindi del prezzo, aggiorna il costo totale dei prodotti del carrello appropriato nella tabella OrdineInCorso.

```
DELIMITER |

DROP TRIGGER IF EXISTS AggiungiAlCarrello |

CREATE TRIGGER AggiungiAlCarrello

AFTER UPDATE ON Carrello
FOR EACH ROW

	BEGIN

		UPDATE OrdineInCorso OIC
		SET PrezzoProdotti = PrezzoProdotti + NEW.Costo - OLD.Costo
		WHERE OIC.ID = NEW.OrdineInCorso;

	END |

DELIMITER ;
```

Trigger 3: Stessa funzione del trigger precedente, solo che aggiorna il costo di OrdineInCorso quando viene aggiunto un prodotto in un carrello.

```
DELIMITER |

DROP TRIGGER IF EXISTS AggiungiAlCarrello |

CREATE TRIGGER AggiungiAlCarrello

AFTER INSERT ON Carrello
FOR EACH ROW

       BEGIN

              UPDATE OrdineInCorso OIC
              SET PrezzoProdotti = PrezzoProdotti + NEW.Costo
              WHERE OIC.ID = NEW.OrdineInCorso;

       END |

DELIMITER ;
```

Trigger 4: Trigger che alla rimozione di un prodotto dal carrello, aggiorna il costo totale nella tabella OrdineInCorso e se l'ordine in corso conteneva solo quel prodotto, elimina l'entry dalla tabella.

```
DELIMITER |

DROP TRIGGER IF EXISTS RimuoviDalCarrello |

CREATE TRIGGER RimuoviDalCarrello

AFTER DELETE ON Carrello
FOR EACH ROW

       BEGIN

              UPDATE OrdineInCorso OIC
              SET PrezzoProdotti = PrezzoProdotti - OLD.Costo
              WHERE OIC.ID = OLD.OrdineInCorso;

              IF (SELECT PrezzoProdotti
                  FROM OrdineInCorso
                  WHERE ID = OLD.OrdineInCorso) = 0

              THEN

                  DELETE FROM OrdineInCorso
                  WHERE ID = OLD.OrdineInCorso;

              END IF;

       END |

DELIMITER ;
```

Trigger 5: Dopo l'inserimento nella tabella OrdineConcluso, aggiorna il costo totale aggiungendo al costo della fattura il costo della spedizione selezionata.

```
DELIMITER |

DROP TRIGGER IF EXISTS AggiungiSpedizione |

CREATE TRIGGER AggiungiSpedizione

BEFORE INSERT ON OrdineConcluso
FOR EACH ROW

	BEGIN

		IF (NEW.Spedizione = 'Express')
		THEN
			SET NEW.CostoFinale = NEW.CostoFinale + 20;
		ELSEIF (NEW.Spedizione = 'Daily')
		THEN
			SET NEW.CostoFinale = NEW.CostoFinale + 50;

		END IF;

	END |

DELIMITER ;
```