Università degli Studi di Modena e Reggio Emilia

Dipartimento di Ingegneria “Enzo Ferrari”

Corso di Laurea Triennale in Ingegneria Informatica

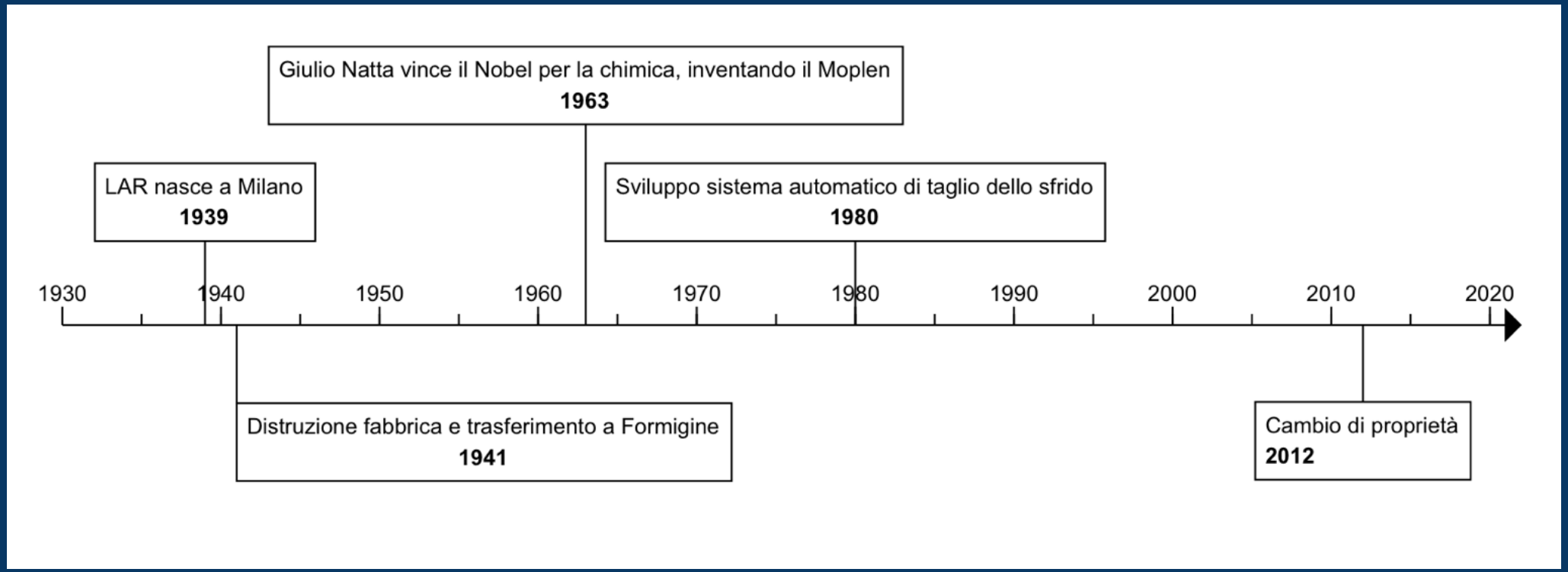
Analisi dati in SQL sul database dell’azienda LAR S.p.A.

Relatore:
Prof. Sonia Bergamaschi

Candidato:
Simone Boschi

Anno Accademico 2019-2020

LAR S.p.A.


Giulio Natta vince il Nobel per la chimica, inventando il Moplen
1963
LAR nasce a Milano
1939
Sviluppo sistema automatico di taglio dello sfrido
1980
1930
1940
1950
1960
1970
1980
1990
2000
2010
2020
Distruzione fabbrica e trasferimento a Formigine
1941
Cambio di proprietà
2012

Obiettivi del tirocinio curriculare

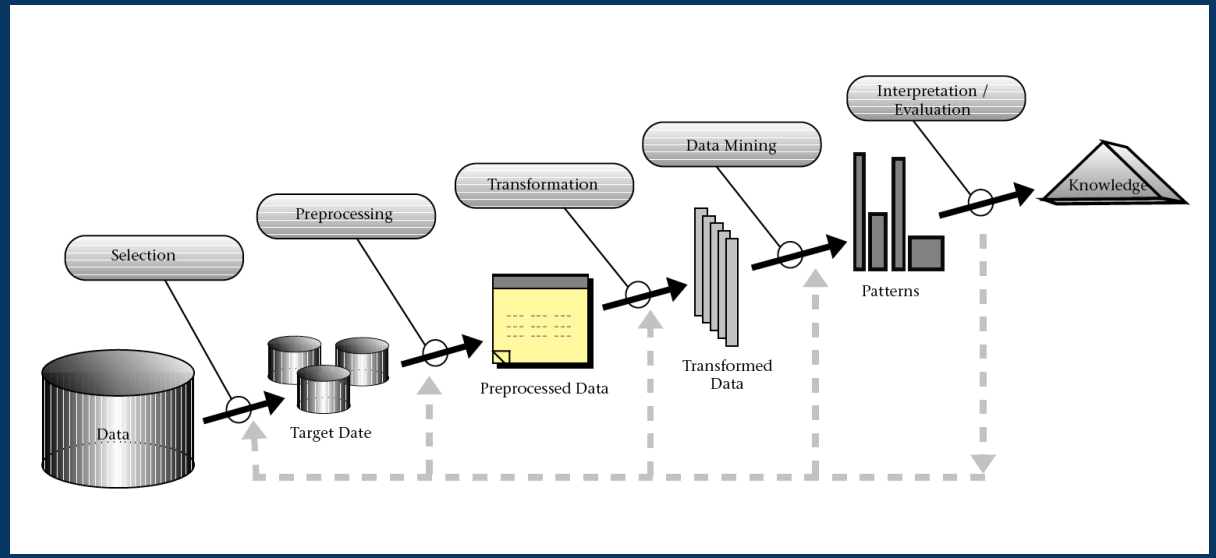
Uno degli obiettivi del tirocinio curriculare era di utilizzare le tecniche di Data Mining per effettuare analisi dei dati.

Sulla base delle mie conoscenze acquisite nel corso di Basi di Dati, ho effettuato le analisi richieste attraverso query in SQL.

Le tecniche di Data Mining possono essere utilizzate per trovare relazioni tra i dati.

Knowledge Discovery in Database (KDD)

1) Selection: selezione dei dati di interesse
2) Preprocessing: rimozione inconsistenze, eliminazione dei casi limite e decisione delle strategie
3) Trasformation: dati trasformati in formati adatti alle tecniche di data mining
4) Data Mining: esecuzione di tecniche per analizzare i dati
5) Interpretation / Evaluation: documentazione e interpretazione dei risultati

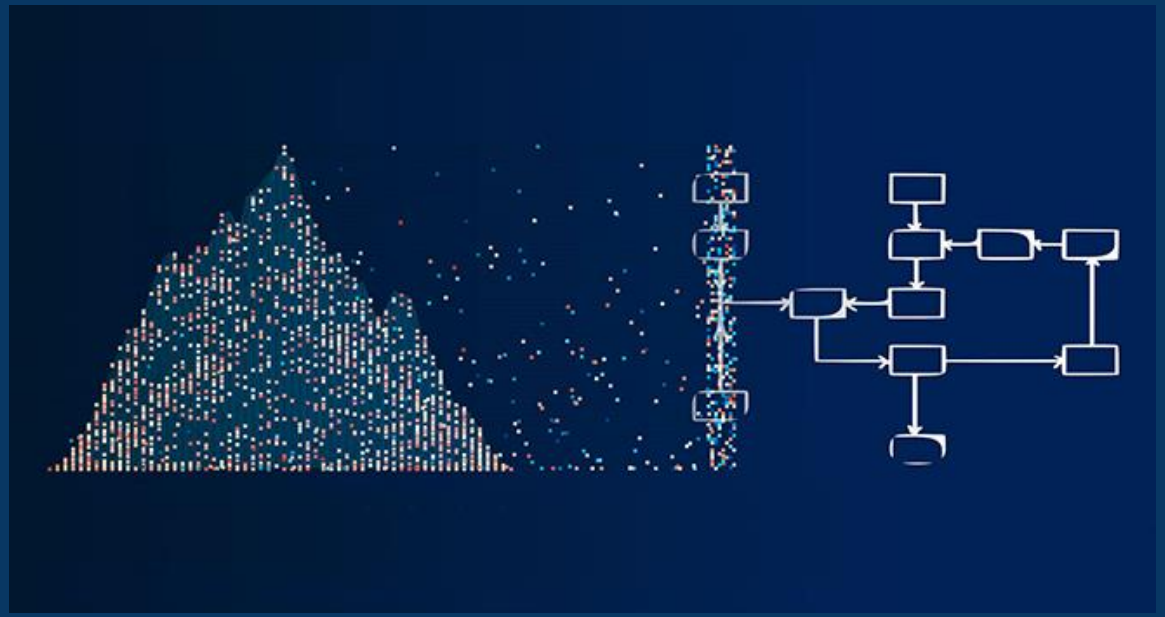


Data Mining

Evoluzione dovuta da due fattori: aumento delle prestazioni dei calcolatori e maggiore quantità di dati a disposizione.

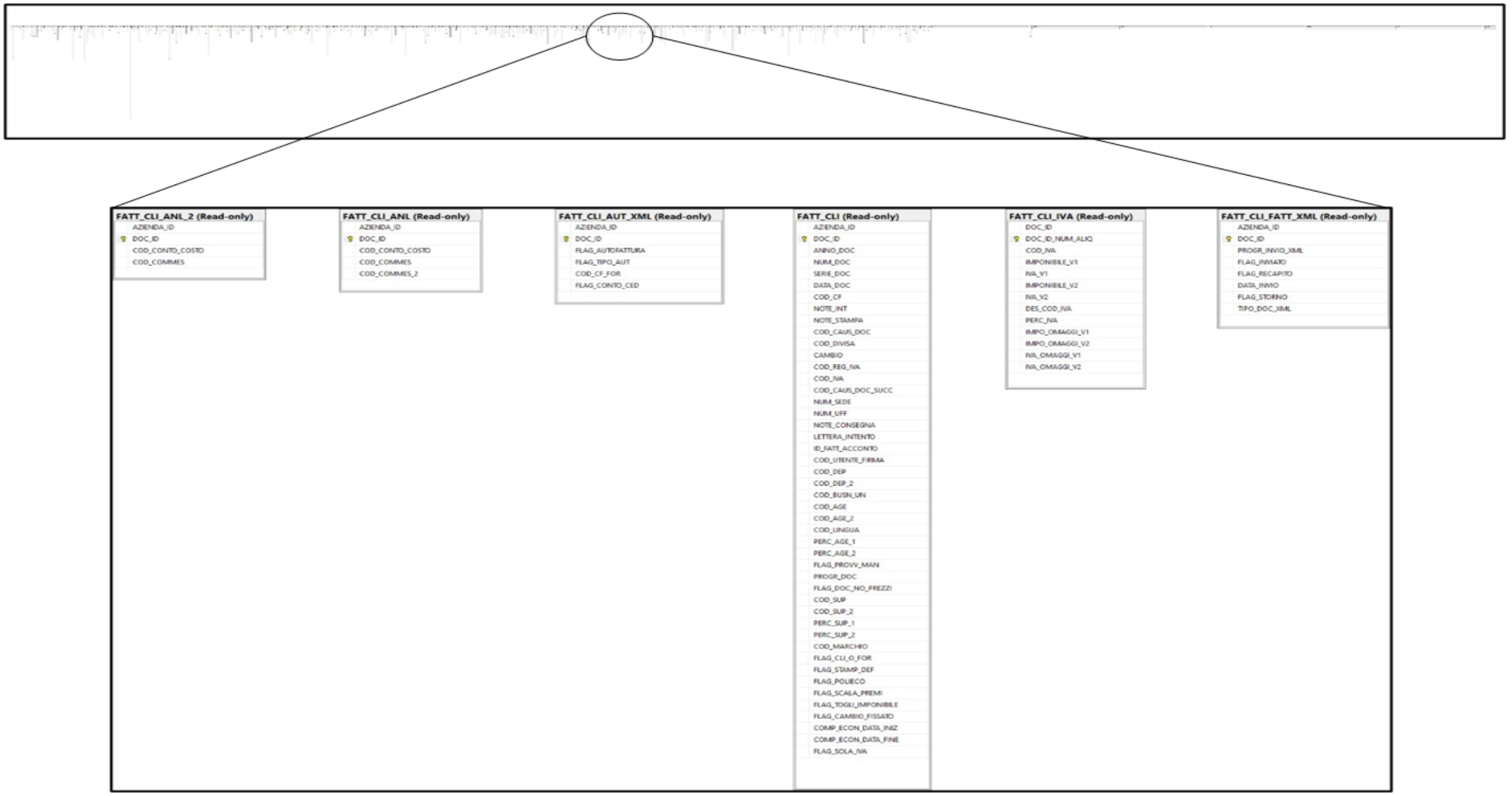
Definizione:

Il data mining è l'insieme di tecniche e metodologie che hanno per oggetto l'estrazione di informazioni utili da grandi quantità di dati attraverso metodi automatici o semi-automatici e l'utilizzo scientifico, aziendale, industriale o operativo delle stesse.



Le principali tecniche:

- Classificazione
- Clustering
- Modellazione delle dipendenze
- Riepilogazione
- Regressione
- Individuazione di deviazioni
- Regole associative

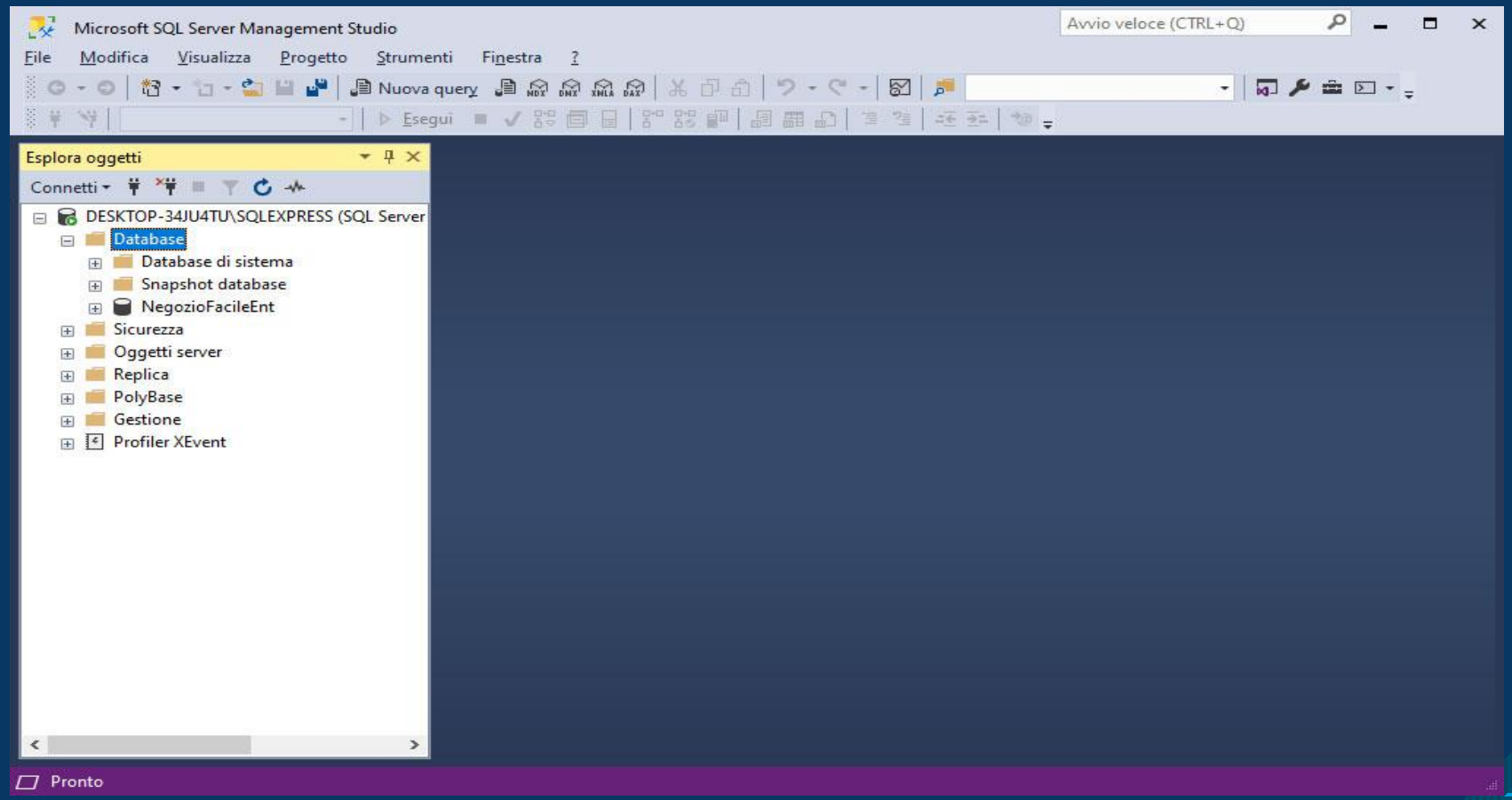

FATT_CLI_ANL_2 (Read-only)
AZIENDA_ID
DOC_ID
COD_CONTO_COSTO
COD_COMMES
FATT_CLI_ANL (Read-only)
AZIENDA_ID
DOC_ID
COD_CONTO_COSTO
COD_COMMES
COD_COMMES_2
FATT_CLI_AUT_XML (Read-only)
AZIENDA_ID
DOC_ID
FLAG_AUTOFATTURA
FLAG_TIPO_AUT
COD_CF_FOR
FLAG_CONTO_CED
FATT_CLI (Read-only)
AZIENDA_ID
DOC_ID
ANNO_DOC
NUM_DOC
SERIE_DOC
DATA_DOC
COD_CF
NOTE_INT
NOTE_STAMPA
COD_CAUS_DOC
COD_DIVISA
CAMBIO
COD_REG_IVA
COD_IVA
COD_CAUS_DOC_SUCC
NUM_SEDE
NUM_UFF
NOTE_CONSEGNA
LETTERA_INTENTO
ID_FATT_ACCONTO
COD_UTENTE_FIRMA
COD_DEP
COD_DEP_2
COD_BUSN_UN
COD_AGE
COD_AGE_2
COD_LINGUA
PERC_AGE_1
PERC_AGE_2
FLAG_PROVV_MAN
PROGR_DOC
FLAG_DOC_NO_PREZZI
COD_SUP
COD_SUP_2
PERC_SUP_1
PERC_SUP_2
COD_MARCHIO
FLAG_CLI_O_FOR
FLAG_STAMP_DEF
FLAG_POLIECO
FLAG_SCALA_PREMI
FLAG_TOGLI_IMPONIBILE
FLAG_CAMBIO_FISSATO
COMP_ECON_DATA_INIZ
COMP_ECON_DATA_FINE
FLAG_SOLA_IVA
FATT_CLI_IVA (Read-only)
DOC_ID
DOC_ID_NUM_ALIQ
COD_IVA
IMPONIBILE_V1
IVA_V1
IMPONIBILE_V2
IVA_V2
DES_COD_IVA
PERC_IVA
IMPO_OMAGGI_V1
IMPO_OMAGGI_V2
IVA_OMAGGI_V1
IVA_OMAGGI_V2
FATT_CLI_FATT_XML (Read-only)
AZIENDA_ID
DOC_ID
PROGR_INVIO_XML
FLAG_INVIATO
FLAG_RECAPITO
DATA_INVIO
FLAG_STORNO
TIPO_DOC_XML

Strumenti utilizzati per l’analisi dei dati

SQL Server Management Studio

DBMS relazionale o RDBMS.

Si basa sul linguaggio SQL.



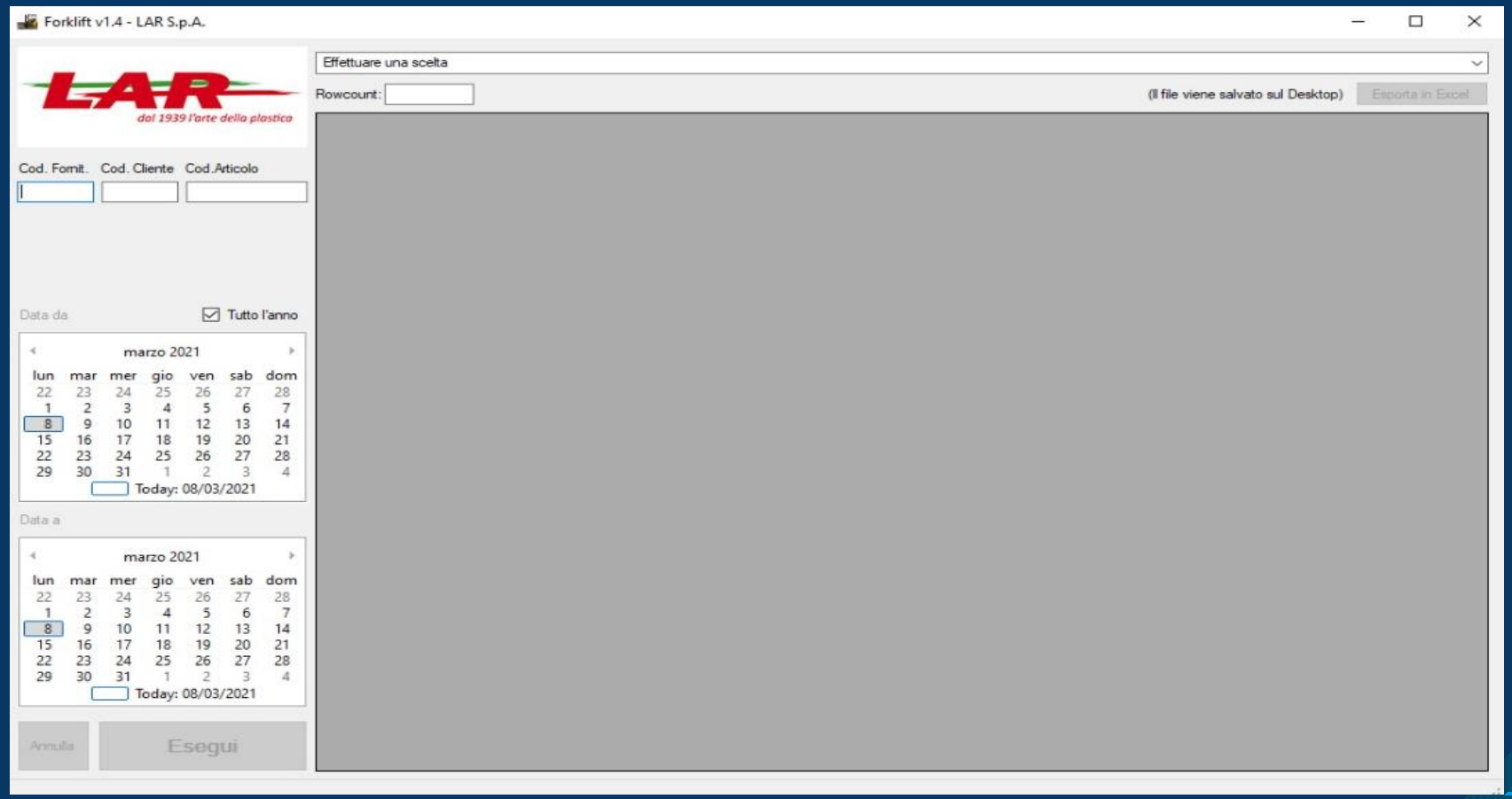
Strumenti utilizzati per l’analisi dei dati

Forklift

Applicazione usata per eseguire le query.

Sviluppata usando:

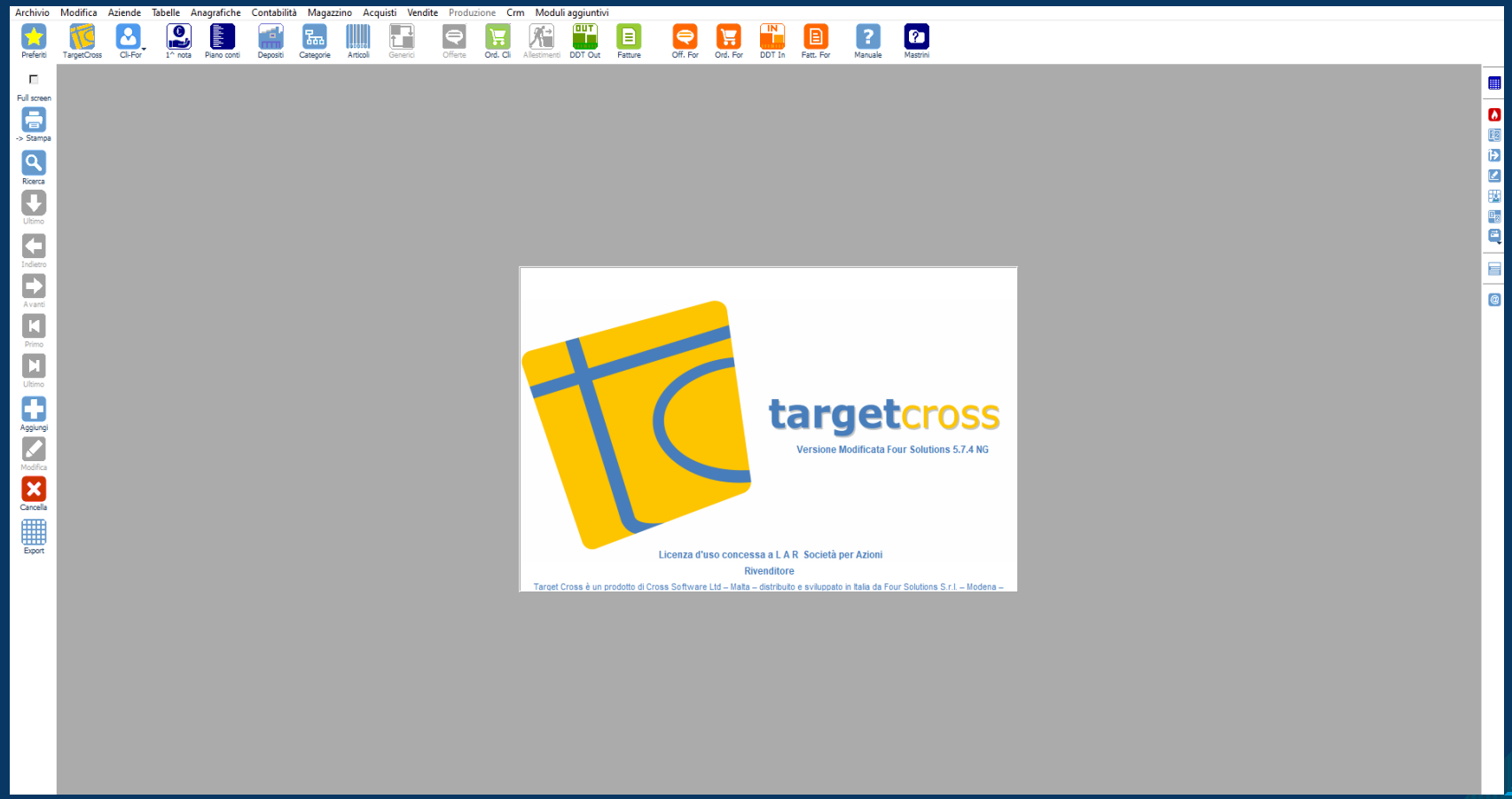
- Visual Studio come ambiente di sviluppo
- .NET come framework
- C# come linguaggio di programmazione



Strumenti utilizzati per l’analisi dei dati

Target Cross

- Applicazione usata per gestire fatture, DDT, ordini, clienti e fornitori.
- Richiesta una connessione al DBMS tramite driver ODBC.
- Presenta una sezione per il CRM.



Query 1 - Clienti che hanno acquistato un determinato articolo nel periodo

```
SELECT DISTINCT
	R.COD_ART AS 'Codice articolo',
	REPLACE(A.DES_ART, CHAR(164), '') AS 'Descrizione articolo',
	CASE WHEN LEN(C.LIV_6_CAT) > 0 THEN C.DESC_L6
	WHEN LEN(C.LIV_5_CAT) > 0 THEN C.DESC_L5
	WHEN LEN(C.LIV_4_CAT) > 0 THEN C.DESC_L4
	WHEN LEN(C.LIV_3_CAT) > 0 THEN C.DESC_L3
	WHEN LEN(C.LIV_2_CAT) > 0 THEN C.DESC_L2
	WHEN LEN(C.LIV_1_CAT) > 0 THEN C.DESC_L1 ELSE ''
	END AS 'Categoria articolo',
	CF2.COD_AGE1_CLI AS 'Codice agente',
	AG.NOME_AGE AS 'Descrizione agente',
	O.COD_CF AS 'Codice cliente',
	CF.RAG_SOC_CF AS 'Descrizione cliente',
	CF.TEL_CF AS 'Telefono',
	CF.E_MAIL_CF AS 'Email',
	CF.PROVINCIA_CF AS 'Provincia'
	CF.STATO_CF AS 'Stato',
	CAST(SUM(R.QUANT_RIGA) OVER(PARTITION BY (R.COD_ART + O.COD_CF))
	AS INT) AS 'Quantità ordinata'

FROM ORD_CLI O
	LEFT JOIN ORD_CLI_RIGHE R ON O.DOC_ID = R.DOC_ID
	LEFT JOIN ART_ANA A ON A.COD_ART = R.COD_ART
	LEFT JOIN CAT_MERCE C ON C.COD_CAT = A.COD_CAT
	LEFT JOIN CF_CLI_4B CF2 ON CF2.COD_CF = O.COD_CF AND CF2.AZIENDA_ID =
	O.AZIENDA_ID
	LEFT JOIN AGENTI AG ON AG.COD_AGE = CF2.COD_AGE1_CLI AND
	AG.AZIENDA_ID = O.AZIENDA_ID
	JOIN CF ON CF.COD_CF = O.COD_CF

WHERE O.AZIENDA_ID = 1
	AND O.DATA_DOC BETWEEN '" + fromDateYDM + "' AND '" + toDateYDM + "'
	-- fromDateYDM e toDateYDM sono due variabili relative alla data

	if (itemCode != string.Empty)  -- this is taken by Visual Studio code and there is a
		AND R.COD_ART LIKE '" + itemCode + "%' -- check with an if statement
	AND A.COD_CAT NOT LIKE 'VIRT%'   -- excludes virtual item
	AND A.COD_CAT NOT LIKE 'ZDG%' -- exclude other non-items

ORDER BY O.COD_CF
```

Obiettivo: Trovare tutti i clienti che hanno acquistato un determinato prodotto in un determinato periodo, oppure lasciando vuoto il campo articolo trovare, dato un periodo, tutti i clienti che hano acquistato dei prodotti e quali di questi.

Campo d'applicazione: Commerciale e produzione.

Input: Data da, Data a e [Cod.Articolo]

Funzioni/Statement utilizzati:

- CASE Statement
- REPLACE()
- CAST()
- SUM()
- OVER([PARTITION BY])

Query 4 - Clienti basso-acquistanti

```
WITH CTE_DateUltimeFatture(COD_CF, UltimaDataAcquisto) AS
(
        SELECT DISTINCT
                F.COD_CF,
                MAX(F.DATA_DOC) OVER(PARTITION BY F.COD_CF) AS
                UltimaDataAcquisto
        FROM FATT_CLI F
        WHERE F.AZIENDA_ID = 1
                AND F.COD_CAUS_DOC LIKE 'FAT[_] [_]C'  -- the _ is the placeholder for a
                single char, we use[_] to escape it as we want to search for FATT_singlechar_C
                AND F.DATA_DOC < '" + toDateYDM + "'
)

SELECT
        F.COD_CF AS 'Codice cliente',
        CF.RAG_SOC_CF AS 'Descrizione cliente',
        CF.P_IVA_CF AS 'Partiva IVA',
        CF.TEL_CF AS 'Telefono',
        CF.E_MAIL_CF AS 'Email',
        R.COD_ART AS 'Codice articolo',
        REPLACE(A.DES_ART, CHAR(164), '') AS 'Descrizione articolo',
        C.LIV_1_CAT AS 'Categoria',
        CAST(R.PREZZO_LORDO_VU1 AS DECIMAL(10, 3)) AS 'Prezzo d''acquisto',
        F.DATA_DOC AS 'Ultima data d''acquisto'
FROM FATT_CLI F
        JOIN CTE_DateUltimeFatture DUF ON DUF.COD_CF = F.COD_CF AND
        DUF.UltimaDataAcquisto = F.DATA_DOC
        JOIN CF ON CF.COD_CF = F.COD_CF
        LEFT JOIN FATT_CLI_RIGHE R ON F.DOC_ID = R.DOC_ID
        LEFT JOIN ART_ANA A ON A.COD_ART = R.COD_ART
        LEFT JOIN CAT_MERCE C ON C.COD_CAT = A.COD_CAT

WHERE F.AZIENDA_ID = 1

        if (itemCode != string.Empty)
                AND R.COD_ART LIKE '" + itemCode + "%'

        AND A.COD_CAT NOT LIKE 'VIRT%'   --excludes virtual items
        AND A.COD_CAT NOT LIKE 'ZDG%'    --exclude other non-items

ORDER BY F.DATA_DOC, F.COD_CF ASC
```

Obiettivo: Verificare i clienti persi, valutando la data in cui hanno acquistato per l'ultima volta un prodotto dall'azienda.

Campo d'applicazione: Commerciale.

Input: Data a e [Cod.Articolo]

Funzioni/Statement utilizzati:

- WITH
- REPLACE()
- CAST Statement
- MAX()

Query 8 - Fatture per variazione prezzo articolo

```
SELECT
	R.COD_CF AS 'Codice cliente',
	CF.RAG_SOC_CF AS 'Descrizione cliente',
	F.NUM_DOC AS 'Numero fattura',
	R.DOC_ID AS 'N. completo',
	R.NUM_RIGA AS 'Riga fattura',
	R.COD_ART AS 'Codice articolo',
	R.DES_RIGA AS 'Descrizione articolo',
	CAST(R.PREZZO_LORDO_VU1 AS DECIMAL(10,3)) AS 'Prezzo unitario',
	R.UM_BASE AS 'UM base',
	R.QUANT_UM_BASE AS 'Quantità base',
	R.UM AS 'UM',
	R.QUANT_RIGA AS 'Quantità',
	CAST(R.IMPORTO_V1 AS DECIMAL(10,2)) AS 'Importo'

FROM FATT_CLI F
	JOIN FATT_CLI_RIGHE R ON F.DOC_ID = R.DOC_ID
	LEFT JOIN CF ON CF.COD_CF = F.COD_CF

WHERE F.AZIENDA_ID = 1
	AND F.DATA_DOC BETWEEN '" + fromDateYDM + "' AND '" + toDateYDM + "' -- The
	BETWEEN operator is inclusive: begin and end values are included.

	if (customerCode != string.Empty)
		AND F.COD_CF = '" + customerCode + "'

	if (itemCode != string.Empty)
		AND R.COD_ART LIKE '" + itemCode + "%'

ORDER BY R.COD_CF, F.NUM_DOC, R.NUM_RIGA
```

Obiettivo: Visualizzare le possibili variazioni del prezzo di un articolo, per tutti i clienti o per uno singolo. Utile anche per analizzare il numero di quantità vendute, in base al prezzo per un determinato articolo

Campo d'applicazione: Amministrativo e commerciale.

Input: Data da, Data a, [Cod.Articolo] e [Cod.Cliente].

Funzioni/Statement utilizzati:

- CAST()

GRAZIE PER L'ATTENZIONE

Query 2 - Totale fatturato in periodo per cliente

```
SELECT DISTINCT
        F.COD_CF AS 'Codice cliente',
        CF.RAG_SOC_CF AS 'Descrizione cliente',
        CF.P_IVA_CF AS 'Partiva IVA',
        CF.TEL_CF AS 'Telefono',
        CF.E_MAIL_CF AS 'Email',
        CF.INDI_CF as 'Indirizzo',
        CF.CAP_CF as 'Cap',
        CF.COMUNE_CF as 'Comune',
        CF.PROVINCIA_CF as 'Provincia',
        CF.STATO_CF as 'Stato',
        CAST(SUM(T.TOTALE_V1) OVER(PARTITION BY F.COD_CF) AS DECIMAL(10, 3))
        AS 'Totale fatturato'

FROM FATT_CLI F
        LEFT JOIN FATT_CLI_TOT T ON T.DOC_ID = F.DOC_ID
        JOIN CF ON CF.COD_CF = F.COD_CF

WHERE F.AZIENDA_ID = 1
        AND F.COD_CAUS_DOC LIKE 'FAT[_]_[_]_'  -- the _ is the placeholder for a single char,
        -- we use[_] to escape it as we want to search for FATT_singlechar_C
        AND F.DATA_DOC BETWEEN '" + fromDateYDM + "' AND '" + toDateYDM + "'

ORDER BY 'Totale fatturato' DESC
```

Obiettivo: Conoscenza dell'importo totale, ovvero la somma delle fatture, di ogni singolo cliente dato un periodo scelto a priori dall'utente.

Campo d'applicazione: Amministrativo.

Input: Data da e Data a.

Funzioni/Statement utilizzati:

- CAST()
- SUM()
- OVER([PARTITION BY])

Query 3 - Quantità prodotta ad 8 settimane da oggi

```
SELECT DISTINCT
        CF.COD_CF AS 'Codice cliente',
        CF.RAG_SOC_CF AS 'Descrizione cliente',
        OP.COD_ART AS 'Codice articolo LAR',
        AC.COD_SECONDARIO_ART AS 'Codice articolo DKT',
        OP.DES_PROD AS 'Descrizione articolo',
        OP.UM AS 'Unità di misura',
        SUM(OP.QUANT_DA_PROD) OVER(PARTITION BY OP.COD_ART) AS 'Qta prodotta
        a + 8 weeks',
        DATEADD(WEEK, 8, GETDATE()) AS 'Data di fine prevista'

FROM ORP_EFF OP
        JOIN CF ON CF.COD_CF = OP.COD_CF
        LEFT JOIN ART_CODICI AC ON AC.COD_ART = OP.COD_ART
        LEFT JOIN ART_ANA A ON A.COD_ART = AC.COD_ART

WHERE OP.AZIENDA_ID = 1
        AND OP.COD_CF = '000009472'
        AND OP.DATA_FINE_PREVISTA <= DATEADD(WEEK, 8, GETDATE())

ORDER BY 4
```

Obiettivo: Verificare la quantità per ogni articolo da produrre o già prodotta, relativa agli ordini con scadenza entro le due mensilità dalla data in cui si esegue questa query.

Campo d'applicazione: Logistica e produzione.

Input: -

Funzioni/Statement utilizzati:

- SUM()
- OVER([PARTITION BY])
- DATEADD()
- GETDATE()

Query 5 - Quantità venduta per articolo

```
WITH CTE_Sum(CodiceArticolo, DescrizioneArticolo, Categoria, Qty) AS
(
	SELECT DISTINCT
		R.COD_ART AS CodiceArticolo,
		REPLACE(A.DES_ART, CHAR(164), '') AS DescrizioneArticolo,
		CASE WHEN LEN(C.LIV_6_CAT) > 0 THEN C.LIV_6_CAT
		WHEN LEN(C.LIV_5_CAT) > 0 THEN C.LIV_5_CAT
		WHEN LEN(C.LIV_4_CAT) > 0 THEN C.LIV_4_CAT
		WHEN LEN(C.LIV_3_CAT) > 0 THEN C.LIV_3_CAT
		WHEN LEN(C.LIV_2_CAT) > 0 THEN C.LIV_2_CAT
		WHEN LEN(C.LIV_1_CAT) > 0 THEN C.LIV_1_CAT ELSE ''
		END AS Categoria,
		CAST(SUM(R.QUANT_RIGA) OVER(PARTITION BY R.COD_ART) AS INT)
		AS Qty

	FROM FATT_CLI_RIGHE R
		JOIN ART_ANA A ON A.COD_ART = R.COD_ART
		LEFT JOIN CAT_MERCE C ON C.COD_CAT = A.COD_CAT

	WHERE A.COD_CAT NOT LIKE 'VIRT%'  --excludes virtual items
		AND A.COD_CAT NOT LIKE 'ZDG%'  --exclude other non-items
		AND R.DATA_DOC BETWEEN '" + fromDateYDM + "' AND '" + toDateYDM +
		"'  -- The BETWEEN operator is inclusive: begin and end values are included.

		if (itemCode != string.Empty)
			AND R.COD_ART LIKE '" + itemCode + "%'
)


SELECT
	S.CodiceArticolo AS 'Codice articolo',
	S.DescrizioneArticolo AS 'Descrizione articolo',
	S.Categoria AS 'Categoria',
	S.Qty AS 'Quantità'

FROM CTE_Sum S

ORDER BY S.Qty DESC
```

Obiettivo: Verificare la quantità venduta di un articolo o di tutti in un determinato periodo, potendo in un successive momento effettuare analisi di mercato e di vendita.

Campo d'applicazione: Commerciale.

Input: Data da, Data a e [Cod.Articolo].

Funzioni/Statement utilizzati:

- WITH
- REPLACE()
- CASE Statement
- CAST()
- SUM()
- OVER ([PARTITION BY])

Query 6 - Previsione esborsi

```
SELECT
	CF.COD_CF AS 'Codice fornitore',
	 CF.RAG_SOC_CF AS 'Descrizione fornitore',
	O.NUM_DOC AS 'Numero ordine',
	SUM(R.IMPORTO_V1) AS 'Totale imponibile',
	RS.DATA_CONS_RIGA AS 'Data consegna',
	C.DES_PAGA AS 'Modalità di pagamento',
	'' AS 'Scadenza pagamento fattura'


 FROM ORD_FOR O
	JOIN CF ON CF.COD_CF = O.COD_CF AND CF.FLAG_FOR = 1
	LEFT JOIN ORD_FOR_RIGHE R ON R.DOC_ID = O.DOC_ID
	LEFT JOIN ORD_FOR_PAG P ON P.DOC_ID = O.DOC_ID
	LEFT JOIN CONDPA C ON C.COD_PAGA = P.COD_PAGA
	LEFT JOIN ORD_FOR_RIGHE_SPEC RS ON RS.DOC_RIGA_ID = R.DOC_RIGA_ID


WHERE O.AZIENDA_ID = 1
	AND RS.DATA_CONS_RIGA BETWEEN '" + fromDateYDM + "' AND '" + toDateYDM
	+ "' -- yyyyddmm - The BETWEEN operator is inclusive: begin and end values are
	included.


GROUP BY CF.COD_CF, CF.RAG_SOC_CF, O.NUM_DOC, C.DES_PAGA,
RS.DATA_CONS_RIGA, R.IMPORTO_V1


ORDER BY O.NUM_DOC
```

Obiettivo: Analizzare dato un periodo di tempo scelto dall'utente tutti gli esborsi aziendali.

Campo d'applicazione: Amministrativo.

Input: Data da e Data a.

Funzioni/Statement utilizzati:

- SUM()

Query 7 - Product Order DKT settimanale/mensile

```
SELECT
	CF.COD_CF AS 'Codice cliente',
	CF.RAG_SOC_CF AS 'Descrizione cliente',
	O.DATA_DOC AS 'Data ordine',
	O.NUM_DOC AS 'Numero ordine LAR',
	OS.NUM_ORDINE_CLIENTE AS 'Numero ordine PO DKT',
	R.COD_ART AS 'Codice articolo LAR',
	AC.COD_SECONDARIO_ART AS 'Codice articolo DKT',
	CAST(R.PREZZO_NETTO_VU1 AS DECIMAL(10, 3)) AS 'Importo EXW',
	R.QUANT_RIGA AS 'Quantità',
	R.UM AS 'UM',
	CAST(SUM(R.PREZZO_NETTO_VU1*R.QUANT_RIGA) OVER(PARTITION BY
	R.COD_ART) AS DECIMAL(10,3)) AS 'Importo totale mensile',
	RS.DATA_CONS_RICH_RIGA AS 'Data consegna',
	DATEPART(MONTH, RS.DATA_CONS_RICH_RIGA) AS 'Mese'

	--DATEPART(WEEK, RS.DATA_CONS_RICH_RIGA) AS 'Settimana consegna'
	-- if we want this query weekly and not monthly

FROM ORD_CLI O
	JOIN CF ON CF.COD_CF = O.COD_CF
	LEFT JOIN ORD_CLI_RIGHE R ON R.DOC_ID = O.DOC_ID
	LEFT JOIN ORD_CLI_RIGHE_SPEC RS ON RS.DOC_RIGA_ID = R.DOC_RIGA_ID
	LEFT JOIN ORD_CLI_SPEC OS ON OS.DOC_ID = O.DOC_ID
	LEFT JOIN ART_CODICI AC ON AC.COD_ART = R.COD_ART
	LEFT JOIN ART_ANA A ON A.COD_ART = AC.COD_ART

WHERE O.COD_CF = '000009472'
	AND R.COD_ART != 'V1'  -- excludes unwanted items
	AND O.DATA_DOC >= ''' + fromDateYDM + '''    -- used for DataOrdine
	AND RS.DATA_CONS_RICH_RIGA <= ''' + toDateYDM + '''  -- used for DataConsegna
	AND 1 = ISNUMERIC(OS.NUM_ORDINE_CLIENTE)  -- excludes
	unwanted/incomplete/canceled orders which are usually prefixed by !!, --, **, or similar

ORDER BY RS.DATA_CONS_RICH_RIGA
```

Obiettivo: Determinare e analizzare il numero di prodotti ordinati mensilmente o settimanalmente da parte del cliente DKT. Utile per effettuare previsioni a livello produttivo e anche logistico.

Campo d'applicazione: Logistica e produzione.

Input: Data da e Data a.

Funzioni/Statement utilizzati:

- CAST()
- SUM()
- OVER([PARTITION BY])
- DATEPART()
- ISNUMERIC()