2019/2020

Progetto di Basi di dati

Progettazione di una base di dati che registra l'andamento del contagio da COVID-19 in Italia

Federica Del Vecchio

N46004430

Indice

1. La creazione di una tabella master in cui inserire i dati del contagio da COVID-19

Lo scopo di questo lavoro è la creazione di una base di dati che mostri l'andamento del contagio da virus COVID-19 per tutte le province italiane dal 25/02/2020 al 03/05/2020. Le informazioni vengono estratte da una serie di file in formato CSV, scaricabili attraverso il link https://github.com/pcm-dpc/COVID-19/tree/master/dati-province, messi a disposizione dal Dipartimento della Protezione Civile.

Per inserire i dati all'interno del database, si crea una tabella master COVID che presenta lo stesso schema dei file CSV (ovvero gli stessi attributi nello stesso ordine). In tal modo, si possono utilizzare strumenti di conversione CSV→SQL, come il tool https://www.convertcsv.com/csv-to-sql.htm, per poter inserire efficacemente le informazioni all'interno della tabella.

In seguito, è riportato lo schema della relazione COVID e la sua creazione attraverso statement DDL, di Data Definition Language, del linguaggio SQL-3.

COVID (Data, Stato, Codice Regione, Denominazione Regione, Codice Provincia, Denominazione Provincia, Sigla Provincia, Latitudine, Longitudine, Totale Casi, Note Ita, Note Eng);

```
CREATE TABLE COVID(
data DATE,
stato CHAR(3),
codice_regione NUMBER(2),
denominazione_regione VARCHAR2(200),
codice_provincia NUMBER(3) NOT NULL,
denominazione_provincia VARCHAR2(200),
sigla_provincia CHAR(2),
latitudine NUMBER,
longitudine NUMBER,
totale_casi INTEGER,
note_it CLOB,
note_en CLOB,
CONSTRAINT PK_COVID PRIMARY KEY (data, codice_provincia)
);
```

Come già anticipato, la CREATE TABLE deve essere seguita da una serie di INSERT. Di seguito ne è mostrato un esempio.

```
INSERT INTO COVID VALUES ('26-Mar-2020', 'ITA', 18, 'Calabria', 079, 'Catanzaro', 'CZ', 38.90597598, 16.59440194, 53, NULL, NULL);
```

La chiave primaria della tabella COVID è formata dagli attributi data e codice_provincia; perciò ogni INSERT riguarda un'unica provincia in un determinato giorno. Risulta, quindi, chiaro che devono essere scritti centinaia e centinaia di INSERT, che per ovvie ragioni si evita di riportare.

Dopo aver inserito tutte le informazioni, risulta necessario apportare una serie di modifiche per il corretto funzionamento della base di dati. Si aggiorna il codice_regione per le province di Trento e Bolzano che altrimenti risulterebbero fare parte della stessa regione, quando invece sono Province autonome. Inoltre, si eliminano che righe con denominazione_provincia = 'In fase di definizione/aggiornamento', le quali non aggiungono alcuna informazione.

```
UPDATE COVID SET codice_regione=98 WHERE denominazione_regione='P.A. Bolzano';
UPDATE COVID SET codice_regione=99 WHERE denominazione_regione='P.A. Trento';
DELETE FROM COVID WHERE denominazione_provincia= 'In fase di
definizione/aggiornamento';
```

2. La verifica della 3NF per lo schema della tabella master e l'eventuale decomposizione

La teoria della normalizzazione ha come scopo quello di progettare basi di dati senza anomalie. Una parte di essa consiste nel definire dei "criteri di qualità" che gli schemi delle relazioni devono soddisfare; questi criteri si concretizzano nelle forme normali.

La prima forma normale, oppure forma atomica, richiede che ogni attributo dello schema relazionale sia di un tipo che abbia dominio atomico; le informazioni di un campo non devono quindi mai essere composte o multivalore. La 1NF, da sola, non elimina alcuna anomalia.

Lo schema della tabella COVID soddisfa la 1NF. Si noti, però, che tutti i DBMS supportano solo domini atomici, quindi la necessità di dimostrare la 1NF risulta in questo caso superflua; diventa però necessaria nel momento in cui si deve progettare una base di dati da zero.

Prima di procedere, si deve però definire il concetto di dipendenza funzionale. La DF può essere descritta brevemente come un vincolo di integrità che definisce legami di tipo funzionale tra gli attributi di una relazione.

Una tabella è in seconda forma normale se è in 1NF e se tutti i suoi attributi non primi sono in dipendenza funzionale dall'intera chiave primaria, mai da una parte di essa. In altre parole, sono ammesse solo dipendenze funzionali complete. In questo caso, la chiave primaria di COVID è rappresentata dalla coppia {data, codice_provincia}. Per verificare che lo schema sia in 2NF, si mostrano tutte le DF nella relazione CODIV.

2.1 {data, codice_provincia} → stato, codice_regione, denominazione_regione, denominazione_provincia, sigla_provincia, latitudine, longitudine, totale_casi, note_it, note_en
2.2 codice_regione → denominazione_regione, stato
2.3 codice_provincia → stato, codice_regione, denominazione_regione, denominazione_provincia, sigla_provincia, latitudine, longitudine

Lo schema non è in 2NF per le DF in 2.3: stato, codice_regione, denominazione_regione, denominazione_provincia, sigla_provincia, latitudine e longitudine oltre a dipendere (funzionalmente) dalla chiave primaria, come mostrato in 2.1, dipendono anche da una sola parte di essa, in particolare da codice_provincia.

Per soddisfare la 2NF, la tabella CODIV deve essere decomposta in una serie di tabelle più piccole che soddisfano le condizioni descritte in precedenza e legate tra loro da vincoli di integrità referenziale.

2.4 PROVINCE (<u>codice_provincia</u>, denominazione_provincia, sigla_provincia, latitudine, longitudine, codice_regione, denominazione_regione, stato)
2.5 COVID_PROVINCE (<u>data</u>, <u>codice_provincia</u>:PROVINCE, totale_casi, note_it, note_en)

Infine, una tabella è in terza forma normale se è in 2NF e se tutti gli attributi non primi sono mutuamente indipendenti, ovvero non vi sono tra di essi delle DF. Alternativamente si può dire che vengono eliminate tutte le dipendenze transitive. Mentre la tabella COVID_PROVINCE in 2.5 è in 3NF, la tabella PROVINCE in 2.4 non lo è a causa delle seguenti DF.

2.6 codice_provincia → codice_regione → denominazione_regione, stato

Viene fatta, quindi, un'ulteriore decomposizione per soddisfare la 3NF. Si creano tre tabelle, ognuna della quali ha un numero minore di attributi rispetto alle precedenti; ed esse sono legate tra loro da vincoli inter-referenziali.

2.7 COVID_PROVINCE(data, codice_provincia:PROVINCE, totale_casi, note_it, note_en)
2.8 REGIONI (codice_regione, denominazione_regione, stato)
2.9 PROVINCE(codice_provincia, denominazione_provincia, sigla_provincia, latitudine, longitudine, codice_regione:REGIONI)

Il processo di normalizzazione descritto finora deve essere poi implementato nella base di dati attraverso classici statement SQL. Di seguito è riportata la scrittura di questo svolgimento.

```
CREATE TABLE REGIONI(
stato CHAR(3),
codice_regione NUMBER(2),
denominazione_regione VARCHAR2(200),
CONSTRAINT PK_REGIONI PRIMARY KEY(codice_regione)
);
INSERT INTO REGIONI SELECT DISTINCT stato, codice_regione, denominazione_regione
FROM COVID ORDER BY codice_regione;
CREATE TABLE PROVINCE(
codice_regione NUMBER(2),
codice_provincia NUMBER(3) NOT NULL,
denominazione_provincia VARCHAR2(200),
sigla_provincia CHAR(2),
latitudine NUMBER,
longitudine NUMBER,
CONSTRAINT PK_PROVINCE PRIMARY KEY(codice_provincia)
);
INSERT INTO PROVINCIE SELECT DISTINCT codice_regione, codice_provincia,
denominazione_provincia, sigla_provincia, latitudine, longitudine FROM COVID ORDER BY
codice_regione;
CREATE TABLE COVID_PROVINCE(
data DATE,
codice_provincia NUMBER(3) NOT NULL,
totale_casi INTEGER,
note_it CLOB,
note_en CLOB,
```

```
CONSTRAINT PK_COVID_PROVINCE PRIMARY KEY(data,codice_provincia)
);
INSERT INTO COVID_PROVINCE SELECT DISTINCT data, codice_provincia, totale_casi,
note_it, note_en FROM COVID ORDER BY data;
```

```
ALTER TABLE PROVINCE ADD CONSTRAINT FK_province FOREIGN KEY(codice_regione)
REFERENCES REGIONI(codice_regione) ON UPDATE CASCADE;
ALTER TABLE COVID_PROVINCE ADD CONSTRAINT FK_covid_province FOREIGN KEY
(codice_provincia) REFERENCES PROVINCE(codice_provincia) ON UPDATE CASCADE;
```

I vincoli di integrità referenziale non sono stati definiti nei CREATE TABLE stessi, ma sono stati aggiunti in seguito con degli ALTER TABLE in modo da non causare anomalie legate all'ordine di definizione delle relazioni. Infatti, quando viene definita una chiave esterna sull'attributo di una tabella è necessario che l'attributo che esso referenzia faccia parte di una tabella già creata.

Poiché non è possibile reperire le informazioni riguardo la diffusione del contagio per ogni singola provincia, viene creata una nuova relazione COVID_REGIONI. Successivamente questa tabella verrà arricchita con una serie di informazioni utili, estratte da https://github.com/pcm-dpc/COVID-19/tree/master/dati-regioni, quali, ad esempio, il numero dei decessi, il numero dei tamponi svolti, il numero di persone ricoverate in terapia intensiva ecc. per ogni regione in un dato giorno. Idealmente queste informazioni sarebbero inserite per ogni provincia e solo successivamente si calcolerebbero i dati per ogni regione, attraverso l'utilizzo di viste e stored procedure, ma per mancanza di risorse si accetta questa "ridondanza" di dati.

```
CREATE TABLE COVID_REGIONI(
data DATE,
codice_regione NUMBER(2),
CONSTRAINT PK_COVID_REGIONI PRIMARY KEY(data,codice_regione)
);
INSERT INTO COVID_REGIONI SELECT DISTINCT data,codice_regione FROM COVID
ORDER BY codice_regione;
```

```
ALTER TABLE COVID_REGIONI ADD CONSTRAINT FK_covid_regione FOREIGN KEY
(codice_regione) REFERENCES REGIONI(codice_regione) ON UPDATE CASCADE;
```

3. L’arricchimento delle tabelle mediante comandi di ALTER TABLE

Ogni tabella creata viene arricchita con delle informazioni utili all’analisi della realtà rappresentata dalla base di dati. Per fare ciò vengono definiti nuovi attributi e vengono inseriti, per ognuno di essi, i valori corrispondenti, attraverso delle istruzioni di ALTER TABLE e di UPDATE TABLE.

```
ALTER TABLE REGIONI ADD numero_abitanti INTEGER;
ALTER TABLE REGIONI ADD superficie NUMBER;
ALTER TABLE REGIONI ADD densita INTEGER;
ALTER TABLE REGIONI ADD posti_letto_terapia_intensiva INTEGER;
ALTER TABLE REGIONI ADD numero_aereoporti INTEGER;
ALTER TABLE REGIONI ADD numero_stazioni INTEGER;
ALTER TABLE REGIONI ADD numero_autostrade INTEGER;
ALTER TABLE REGIONI ADD numero_strade_statali INTEGER;
```

```
ALTER TABLE PROVINCE ADD numero_abitanti INTEGER;
ALTER TABLE PROVINCE ADD numero_comuni INTEGER;
ALTER TABLE PROVINCE ADD superficie NUMBER;
ALTER TABLE PROVINCE ADD densita INTEGER;
ALTER TABLE PROVINCE ADD numero_scuole INTEGER;
ALTER TABLE PROVINCE ADD numero_ospedali INTEGER;
```

```
ALTER TABLE COVID_REGIONI ADD ricoverati_con_sintomi INTEGER;
ALTER TABLE COVID_REGIONI ADD terapia_intensiva INTEGER;
ALTER TABLE COVID_REGIONI ADD totale_ospedalizzati INTEGER;
ALTER TABLE COVID_REGIONI ADD isolamento_domiciliare INTEGER;
ALTER TABLE COVID_REGIONI ADD variazione_totale_positivi INTEGER;
ALTER TABLE COVID_REGIONI ADD totale_positivi INTEGER;
ALTER TABLE COVID_REGIONI ADD casi_totali INTEGER;
ALTER TABLE COVID_REGIONI ADD nuovi_positivi INTEGER;
ALTER TABLE COVID_REGIONI ADD dimessi_guariti INTEGER;
ALTER TABLE COVID_REGIONI ADD deceduti INTEGER;
ALTER TABLE COVID_REGIONI ADD tamponi INTEGER;
ALTER TABLE COVID_REGIONI ADD casi_testati INTEGER;
```

Come già anticipato, questi statement sono poi seguiti da una serie di UPDATE TABLE. Di seguito sono riportati degli esempi.

```
UPDATE REGIONI SET numero_abitanti=1947131, superficie=15221.90, densita=128,
posti_letto_terapia_intensiva=206 WHERE codice_regione=18;
UPDATE REGIONI SET numero_aereoporti=5, numero_stazioni=41, numero_autostrade=2,
numero_strade_statali=49 WHERE codice_regione=18;
```

```
UPDATE PROVINCE SET numero_abitanti=308052, superficie=1954.38, densita=158,
numero_comuni=47 WHERE codice_provincia=067;
UPDATE PROVINCE SET numero_scuole=354, numero_ospedali=1 WHERE
codice_provincia=067;
```

```
UPDATE COVID_REGIONI SET ricoverati_con_sintomi=11914, terapia_intensiva=1343,
totale_ospedalizzati=13257, isolamento_domiciliare=15212, totale_positivi=28469,
variazione_totale_positivi=345, nuovi_positivi=1079, dimessi_guariti=13863, deceduti=9202,
casi_totali=51534, tamponi=154989, casi_testati=NULL WHERE data='06-Apr-2020' AND
codice_regione=3;
```

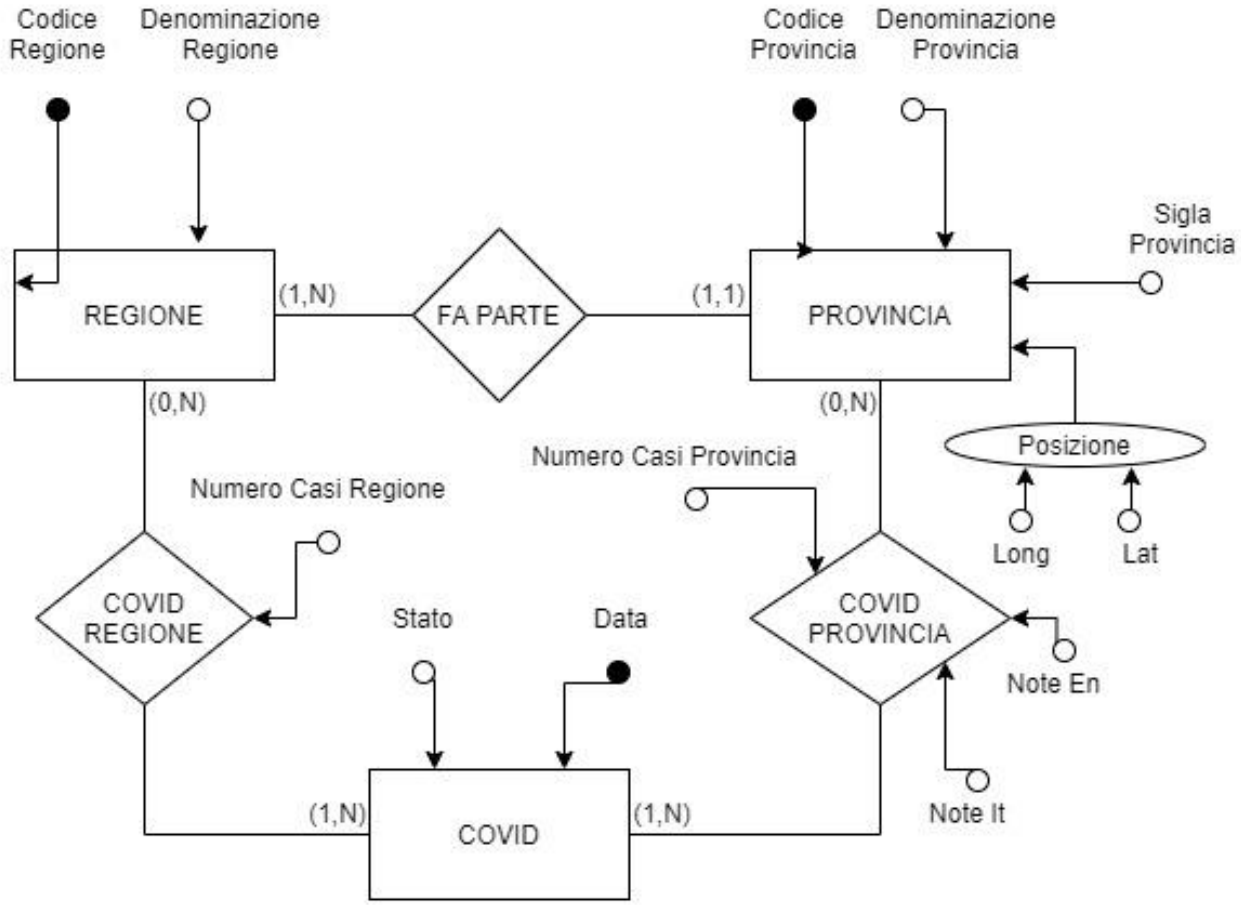
Le fonti dei nuovi dati inseriti sono elencate, sotto forma di commenti, nella scrittura del programma.

4. Lo schema concettuale E/R della base di dati

Attraverso un processo di reverse-engineering, ricaviamo lo schema concettuale della base di dati a partire dagli schemi logico-relazionali descritti al punto 2.

La progettazione concettuale di una base di dati consiste nell'individuare gli oggetti, ovvero le entità che la costituiscono, le proprietà, gli attributi, di ogni oggetto, e le relazioni, le operazioni e le associazioni, tra un oggetto e l'altro.

Un possibile schema E/R che rappresenta la base di dati è il seguente.



- Le entità sono rappresentate dalle realtà REGIONE, PROVINCIA e COVID. Un'entità è una collezione di occorrenze di entità, ovvero concetti che esprimono oggetti realmente esistenti, fisici o astratti.
- Ogni entità è caratterizzata da una serie di proprietà, gli attributi; nell'immagine sono rappresentati solo quelli più importanti, ereditati dalla tabella originale COVID. Per l'entità REGIONE l'attributo identificatore, il cui valore è sempre non nullo e unico per ogni occorrenza, è Codice Regione, per PROVINCIA è Codice Provincia, e per COVID è Data.

L’entità COVID non presenta molti attributi poiché questa non verrà tradotta in una relazione nel modello relazionale della base di dati: avere una terza tabella, oltre COVID_PROVINCE e COVID_REGIONI, che descrive i dati del contagio in Italia risulta essere particolarmente costoso soprattutto perché i dati giornalieri riguardanti l’Italia intera sono facilmente calcolabili a partire dalle tabelle precedenti, attraverso, ad esempio, l’utilizzo di viste.

- Le associazioni che legano tra di loro le entità sono tre. La prima è tra REGIONE e PROVINCIA ed è un’associazione del tipo uno a molti poiché in una regione ci possono essere una o più province e una provincia si trova in una e una sola regione. Un’altra associazione si ha tra REGIONE e COVID, chiamata COVID REGIONE, ed è una associazione di tipo molti a molti poiché il virus COVID-19 si può trovare in una o più regioni e in una regione vi possono essere da zero a N casi di COVID-19. Tale associazione presenta inoltre un attributo proprio che chiamiamo Numero Casi Regione; per la natura stessa degli attributi delle associazioni, ogni occorrenza di Numero Casi Regione riguarda una sola regione in un determinato giorno. L’ultima associazione è quella tra PROVINCIA e COVID, ovvero COVID PROVINCIA. Come prima è una associazione di tipo molti a molti perché il virus si può trovare in una o più province e in una provincia possiamo avere da zero a N casi di COVID-19. Anche questa associazione presenta un attributo locale, Numero Casi Provincia, che opera proprio come Numero Casi Regione.

Consideriamo che il virus sia in almeno una regione e una provincia poiché la base di dati memorizza informazioni a partire dal giorno in cui è stato trovato il primo caso.

Si noti, inoltre, che dal lato PROVINCIA, la cardinalità dell’associazione è (1,1). In realtà le Province autonome di Trento e Bolzano non fanno parte di alcuna regione ma per la natura dell’associazione FA PARTE (e per come essa verrà tradotta nel modello relazionale) si deve comunque assegnare loro un codice_regione. Per ovviare al problema se ne assegnano due diversi.

Quindi si riscrive il modello E/R in modello relazionale, attraverso la progettazione logica, e si confronta con la struttura della base di dati già definita.

Prima di tutto si deve fare un processo di trasformazione, dal modello E/R a uno semplificato che non presenti costrutti non traducibili nel modello relazionale, come gli attributi composti e quelli multivalori. Nell’immagine rappresentata, l’unico attributo non atomico è Posizione in PROVINCIA; esso è attributo composto formato a sua volta da Latitudine e Longitudine. Per risolvere il problema, si cancella Posizione, e Longitudine e Latitudine diventano due attributi semplici di PROVINCIA. Nel caso ci fosse stato un attributo multivalore, questo sarebbe stato sostituito con una nuova entità legata alla prima da una associazione di tipo uno a molti.

Solo a questo punto è possibile effettuare la traduzione effettiva del modello E/R in modello relazionale.

- Ogni entità si traduce in una relazione con lo stesso nome, ma al plurale, e gli stessi attributi. Gli attributi identificatori formano la chiave primaria della relazione.
- Le associazioni uno a molti, come FA PARTE, si eliminano aggiungendo alla relazione che traduce l’entità dal lato uno gli attributi dell’associazione e gli identificatori del lato molti,

eventualmente ridenominati. Esiste un vincolo di identità referenziale tra questi ultimi e la relazione che traduce l'entità dal lato molti.

- Le associazioni molti a molti, come COVID REGIONE e COVID PROVINCIA, si traducono in delle relazioni con lo stesso nome, ma al plurale, gli stessi attributi e gli identificatori delle entità coinvolte; quest'ultimi formano la chiave primaria. Ovviamente esiste un vincolo di integrità referenziale tra essi e le relazioni che traducono le entità coinvolte.

Alla fine, risultano i seguenti schemi relazionali.

REGIONI (Codice Regione, Denominazione Regione);
PROVINCE (Codice Provincia, Denominazione Provincia, Sigla Provincia, Regione Codice: REGIONI, Latitudine, Longitudine);
COVID (Data, Stato);
COVID REGIONI (Cod Regione: REGIONI, Data1: COVID, Numero Casi Regione);
COVID PROVINCE (Cod Provincia: PROVINCE, Data2: COVID, Numero Casi Provincia, Note It, Note en);

È stato già anticipato che questa non rappresenta la struttura finale della base di dati poiché che non esiste una tabella COVID, che descrive la propagazione del contagio per tutta l'Italia. Quindi, si inseriscono le date dei diversi giorni presi in considerazione direttamente in COVID_REGIONI e COVID_PROVINCE, eliminando i vincoli inter-referenziali con COVID. Infine, per non perdere l'informazione Stato, l'attributo viene inserito in COVID_REGIONI, nonostante la ridondanza.

5 La specifica in SQL di una serie di query utili all'analisi dell'andamento del contagio

Con il termine query si intende l'interrogazione fatta da un utente o un'applicazione su un database, per ottenere determinate informazioni.

Rappresentiamo in seguito una serie di query, e i corrispondenti risultati, che si possono fare sulla base di dati. Per alcune query è riportato solo parte del risultato, dato che esso può essere formato anche centinaia di righe.

1. Si seleziona i casi totali e i nuovi positivi per ogni giorno, per la regione oppure le regioni con il maggior numero di aeroporti.

```
SELECT R.denominazione_regione, R.numero_aereoporti, C.data, C.casi_totali,
C.nuovi_positivi
FROM COVID_REGIONI C JOIN REGIONI R ON C.codice_regione=R.codice_regione
WHERE numero_aereoporti=(
SELECT MAX(R2.numero_aereoporti) FROM REGIONI R2)
ORDER BY C.data, R.denominazione_regione;
```

1	DENOMINAZIONE_REGIONE	NUMERO_AEROPORTI	DATA	CASI_TOTALI	NUOVI_POSITIVI
2	Emilia-Romagna	14	2/25/2020	26	8
3	Lombardia	14	2/25/2020	240	68
4	Emilia-Romagna	14	2/26/2020	47	21
5	Lombardia	14	2/26/2020	258	18
6	Emilia-Romagna	14	2/27/2020	97	50
7	Lombardia	14	2/27/2020	403	145
8	Emilia-Romagna	14	2/28/2020	145	48
9	Lombardia	14	2/28/2020	531	128
10	Emilia-Romagna	14	2/29/2020	217	72
11	Lombardia	14	2/29/2020	615	84
12	Emilia-Romagna	14	3/1/2020	285	68
13	Lombardia	14	3/1/2020	984	369
14	Emilia-Romagna	14	3/2/2020	335	50
15	Lombardia	14	3/2/2020	1254	270
16	Emilia-Romagna	14	3/3/2020	420	85
17	Lombardia	14	3/3/2020	1520	266

2. Per ogni regione, si calcola il numero di giorni in cui vi sono stati meno di 50 nuovi positivi.

```
SELECT C.codice_regione, R.denominazione_regione, COUNT(C.data) AS Risultato_query
FROM COVID_REGIONI C JOIN REGIONI R ON C.codice_regione=R.codice_regione
WHERE (C.nuovi_positivi)<=50
GROUP BY C.codice_regione, R.denominazione_regione
ORDER BY codice_regione;
```

1	CODICE_REGIONE	DENOMINAZIONE_REGIONE	RISULTATO_QUERY
2	1	Piemonte	14
3	2	Valle d'Aosta	64
4	3	Lombardia	1
5	5	Veneto	8
6	6	Friuli Venezia Giulia	42
7	7	Liguria	16
8	8	Emilia-Romagna	5
9	9	Toscana	16
10	10	Umbria	56
11	11	Marche	24
12	12	Lazio	17
13	13	Abruzzo	40
14	14	Molise	69
15	15	Campania	33
16	16	Puglia	31
17	17	Basilicata	69
18	18	Calabria	65
19	19	Sicilia	42
20	20	Sardegna	62
21	98	P.A. Bolzano	47
22	99	P.A. Trento	28

3. Si seleziona le informazioni legate al contagio da COVID-19, solo per quelle regioni con più di 2.000.000 di abitanti.

```
SELECT C.*
FROM COVID_REGIONI C
WHERE  2000000 < (
SELECT R.numero_abitanti
FROM REGIONI R
WHERE R.codice_regione = C.codice_regione)
ORDER BY C.data, C.codice_regione;
```

1	DATA	CODICE_REGIONE	RICOV	TERAP	TOTAL	ISOLAM	VARIA	TOTALE	CASI_TC	NUOV	DIMES	DECED	TAMPC
152	3/12/2020	15	56	11	67	107	25	174	179	25	4	1	1551
153	3/12/2020	16	58	2	60	38	27	98	104	27	1	5	1269
154	3/12/2020	19	28	5	33	78	30	111	115	32	2	2	1477
155	3/13/2020	1	556	135	691	103	240	794	840	260	0	46	3105
156	3/13/2020	3	4435	650	5085	2647	836	7732	9820	1095	1198	890	32700
157	3/13/2020	5	366	107	473	980	156	1453	1595	211	100	42	25691
158	3/13/2020	8	942	128	1070	941	253	2011	2263	316	51	201	8787
159	3/13/2020	9	134	77	211	244	103	455	470	106	10	5	4049
160	3/13/2020	12	122	24	146	96	70	242	277	77	24	11	6491
161	3/13/2020	15	60	19	79	134	39	213	220	41	5	2	1671
162	3/13/2020	16	77	2	79	42	23	121	129	25	3	5	1449
163	3/13/2020	19	37	7	44	82	15	126	130	15	2	2	1950
164	3/14/2020	1	538	150	688	126	20	814	873	33	0	59	3680
165	3/14/2020	3	4898	732	5630	3429	1327	9059	11685	1865	1660	966	37138
166	3/14/2020	5	366	119	485	1290	322	1775	1937	342	107	55	26980
167	3/14/2020	8	1076	152	1228	1121	338	2349	2644	381	54	241	10043
168	3/14/2020	9	160	87	247	367	159	614	630	160	10	6	4595
169	3/14/2020	12	181	25	206	114	78	320	357	80	24	13	7335
170	3/14/2020	15	72	17	89	154	30	243	272	52	23	6	1936
171	3/14/2020	16	91	6	97	59	35	156	166	37	2	8	1681
172	3/14/2020	19	42	11	53	97	24	150	156	26	4	2	2100
173	3/15/2020	1	726	171	897	133	216	1030	1111	238	0	81	4375
174	3/15/2020	3	5500	767	6267	3776	984	10043	13272	1587	2011	1218	40369
175	3/15/2020	5	426	129	555	1434	214	1989	2172	235	120	63	32546
176	3/15/2020	8	1215	169	1384	1357	392	2741	3093	449	68	284	12054
177	3/15/2020	9	175	107	282	481	149	763	781	151	10	8	5132
178	3/15/2020	12	223	31	254	142	76	396	436	79	24	16	8345
179	3/15/2020	15	73	22	95	201	53	296	333	61	28	9	2213
180	3/15/2020	16	116	6	122	90	56	212	230	64	2	16	2017

4. A partire dal 15 Aprile, si seleziona le regioni e i giorni con nuovi positivi=0.

```
SELECT R.codice_regione, R.denominazione_regione, C.data, C.nuovi_positivi
FROM COVID_REGIONI C JOIN REGIONI R ON C.codice_regione=R.codice_regione
WHERE nuovi_positivi=0 AND nuovi_positivi IS NOT NULL AND data>='15-Apr -2020'
ORDER BY R.codice_regione, R.denominazione_regione;
```

1	CODICE	DENOMINAZIO	DATA	NUOVI_POS
2	2	Valle d'Aosta	4/20/2020	0
3	2	Valle d'Aosta	4/25/2020	0
4	10	Umbria	5/3/2020	0
5	14	Molise	4/27/2020	0
6	14	Molise	4/29/2020	0
7	14	Molise	5/3/2020	0
8	14	Molise	4/23/2020	0
9	14	Molise	4/16/2020	0
10	14	Molise	4/18/2020	0
11	17	Basilicata	4/29/2020	0
12	17	Basilicata	4/28/2020	0
13	17	Basilicata	4/20/2020	0
14	17	Basilicata	4/27/2020	0
15	18	Calabria	5/2/2020	0

5. Si selezionano i giorni con il totale dei positivi >10.000 e <50.000.

```
SELECT data, SUM(totale_positivi) AS Tot_Positivi
FROM COVID_REGIONI
GROUP BY data
HAVING SUM(totale_positivi)>=10000 AND SUM(totale_positivi)<=50000
ORDER BY data;
```

1	DATA	TOT_POSITIVI
2	3/11/2020	10590
3	3/12/2020	12839
4	3/13/2020	14955
5	3/14/2020	17750
6	3/15/2020	20603
7	3/16/2020	23073
8	3/17/2020	26062
9	3/18/2020	28710
10	3/19/2020	33190
11	3/20/2020	37860
12	3/21/2020	42681
13	3/22/2020	46638

6. Si seleziona la media dei dimessi, nel mese di Aprile, per ogni regione del centro Italia.

```
SELECT C.codice_regione, R.denominazione_regione, AVG(C.dimessi_guariti) AS
Media_dimessi
FROM COVID_REGIONI C JOIN REGIONI R ON C.codice_regione=R.codice_regione
WHERE data>='1-Apr-2020' AND data<='30-Apr-2020' AND
(R.denominazione_regione= 'Lazio' OR R.denominazione_regione= 'Marche' OR
R.denominazione_regione= 'Toscana' OR R.denominazione_regione= 'Umbria')
GROUP BY C.codice_regione, R.denominazione_regione;
```

1	CODICE_REGIONE	DENOMINAZIONE_	MEDIA_DIMESSI
2	9	Toscana	1062,27
3	10	Umbria	647,067
4	12	Lazio	880
5	11	Marche	1255,27

7. Si seleziona per ogni giorno la provincia con il numero maggiore di casi totali.

```
SELECT C1.data, C1.codice_provincia AS Codice_Provincia_con_picco,
P.denominazione_provincia AS Provincia_con_picco, C1.totale_casi AS Picco_di_casi_totali
FROM COVID_PROVINCE C1 JOIN PROVINCE P ON C1.codice_provincia=
P.codice_provincia
WHERE C1.totale_casi= (
SELECT MAX(C2.totale_casi)
FROM COVID_PROVINCE C2
WHERE C2.data=C1.data
GROUP BY C2.data)
ORDER BY C1.data;
```

1	DATA	CODICE_PROVINCIA_CON_	PROVINCI	PICCO_DI_CASI
8	3/2/2020	98	Lodi	384
9	3/3/2020	98	Lodi	482
10	3/4/2020	98	Lodi	559
11	3/5/2020	98	Lodi	658
12	3/6/2020	98	Lodi	739
13	3/7/2020	98	Lodi	811
14	3/8/2020	16	Bergamo	997
15	3/9/2020	16	Bergamo	1245
16	3/10/2020	16	Bergamo	1472
17	3/11/2020	16	Bergamo	1815
18	3/12/2020	16	Bergamo	2136
19	3/13/2020	16	Bergamo	2368
20	3/14/2020	16	Bergamo	2864
21	3/15/2020	16	Bergamo	3416
22	3/16/2020	16	Bergamo	3760
23	3/17/2020	16	Bergamo	3993
24	3/18/2020	16	Bergamo	4305
25	3/19/2020	16	Bergamo	4645
26	3/20/2020	16	Bergamo	5154
27	3/21/2020	16	Bergamo	5869
28	3/22/2020	16	Bergamo	6216
29	3/23/2020	16	Bergamo	6471
30	3/24/2020	16	Bergamo	6728
31	3/25/2020	16	Bergamo	7072
32	3/26/2020	16	Bergamo	7458
33	3/27/2020	16	Bergamo	8060
34	3/28/2020	16	Bergamo	8349
35	3/29/2020	16	Bergamo	8527
36	3/30/2020	15	Milano	8676

8. Si selezionano le informazioni memorizzare in COVID_REGIONI per la regione con la maggiore densità di popolazione, per l'intervallo di tempo tra il 26 Febbraio e il 15 Marzo.

```
SELECT R.denominazione_regione, C.*
FROM COVID_REGIONI C JOIN REGIONI R ON R.codice_regione=C.codice_regione
WHERE R.densita=(SELECT MAX(R1.densita) FROM REGIONI R1) AND C.data between
'26-Feb-2020' AND '15-Mar-2020'
ORDER BY C.data;
```

1	DENOMIN	DATA	CODIC	RICOVE	TERAP	TOTA	ISOLA	VARIA	TOTA	CASI_	NUOVI	DIME	DEC	TAMP(	CASI
2	Campania	2/26/2020	15	0	0	0	0	0	0	0	0	0	0	10	
3	Campania	2/27/2020	15	2	0	2	1	3	3	3	3	0	0	10	
4	Campania	2/28/2020	15	2	0	2	2	1	4	4	1	0	0	213	
5	Campania	2/29/2020	15	3	0	3	10	9	13	13	9	0	0	373	
6	Campania	3/1/2020	15	4	0	4	13	4	17	17	4	0	0	373	
7	Campania	3/2/2020	15	4	0	4	13	0	17	17	0	0	0	373	
8	Campania	3/3/2020	15	11	0	11	19	13	30	30	13	0	0	405	
9	Campania	3/4/2020	15	11	0	11	20	1	31	31	1	0	0	429	
10	Campania	3/5/2020	15	12	0	12	33	14	45	45	14	0	0	471	
11	Campania	3/6/2020	15	12	0	12	45	12	57	57	12	0	0	471	
12	Campania	3/7/2020	15	16	0	16	45	4	61	61	4	0	0	612	
13	Campania	3/8/2020	15	30	7	37	63	39	100	101	40	1	0	980	
14	Campania	3/9/2020	15	42	8	50	69	19	119	120	19	1	0	980	
15	Campania	3/10/2020	15	33	8	41	85	7	126	127	7	1	0	1141	
16	Campania	3/11/2020	15	56	11	67	82	23	149	154	27	4	1	1375	
17	Campania	3/12/2020	15	56	11	67	107	25	174	179	25	4	1	1551	
18	Campania	3/13/2020	15	60	19	79	134	39	213	220	41	5	2	1671	
19	Campania	3/14/2020	15	72	17	89	154	30	243	272	52	23	6	1936	
20	Campania	3/15/2020	15	73	22	95	201	53	296	333	61	28	9	2213	

9. Si calcola la percentuale di ricoverati non gravi, di persone in terapia intensiva e di persone in isolamento domiciliare rispetto al totale dei positivi.

```
SELECT data, SUM(ricoverati_con_sintomi)*100/SUM(totale_positivi) AS
Percentuale_non_gravi, SUM(terapia_intensiva)*100/SUM(totale_positivi) AS
Percentuale_in_terapia_intensiva, SUM(isolamento_domiciliare)*100/SUM(totale_positivi) AS
Percentuale_a_casa
FROM COVID_REGIONI
GROUP BY data
ORDER BY data;
```

1	DATA	PERCENTUALE_NON	PERCENTUALE_TERAPIA	PERCENTUALE_A_CASA
2	2/25/2020	36,65594855	11,25401929	52,09
3	2/26/2020	33,24675325	9,350649351	57,4026
4	2/27/2020	42,17687075	9,523809524	48,2993
5	2/28/2020	42,02192448	7,795371498	50,1827
6	2/29/2020	38,22688275	10,00953289	51,7636
7	3/1/2020	40,51997464	8,877615726	50,6024
8	3/2/2020	40,4359673	9,046321526	50,5177
9	3/3/2020	45,69155988	10,11931065	44,1891
10	3/4/2020	49,74131559	10,90169993	39,357
11	3/5/2020	54,30825243	10,64927184	35,0425
12	3/6/2020	61,13381001	11,79775281	27,0684
13	3/7/2020	52,38095238	11,2033195	36,4157
14	3/8/2020	55,69124785	10,17692187	34,1318
15	3/9/2020	54,05134627	9,17971196	36,7689
16	3/10/2020	59,17312661	10,30068123	30,5262
17	3/11/2020	55,12747875	9,70727101	35,1653
18	3/12/2020	51,79531116	8,980450191	39,2242
19	3/13/2020	49,65563357	8,879973253	41,4644
20	3/14/2020	47,16619718	8,552112676	44,2817
21	3/15/2020	46,90093676	8,115323011	44,9837
22	3/16/2020	47,78312313	8,022363802	44,1945
23	3/17/2020	49,47433044	7,904228378	42,6214
24	3/18/2020	50,02786486	7,861372344	42,1108

10. Si selezionano le informazioni legate al contagio da COVID-19 per la provincia più a nord, per quella più a sud, quella più a est e quella più a ovest.

```
SELECT P.denominazione_provincia, C.*
FROM COVID_PROVINCE C JOIN PROVINCE P ON C.codice_provincia=P.codice_provincia
WHERE P.latitudine IN (SELECT MAX(P1.latitudine) FROM PROVINCE P1)
OR P.latitudine IN (SELECT MIN(P2.latitudine) FROM PROVINCE P2)
OR P.longitudine IN (SELECT MAX(P3.longitudine) FROM PROVINCE P3)
OR P.longitudine IN (SELECT MIN(P4.longitudine) FROM PROVINCE P4)
ORDER BY C.data;
```

1	DENOMINAZIONE	DATA	CODICE	TOTALE_CASI
58	Aosta	3/10/2020	7	17
59	Bolzano	3/10/2020	21	38
60	Lecce	3/10/2020	75	10
61	Ragusa	3/10/2020	88	1
62	Aosta	3/11/2020	7	20
63	Bolzano	3/11/2020	21	75
64	Lecce	3/11/2020	75	12
65	Ragusa	3/11/2020	88	1
66	Aosta	3/12/2020	7	27
67	Bolzano	3/12/2020	21	104
68	Lecce	3/12/2020	75	18
69	Ragusa	3/12/2020	88	2
70	Aosta	3/13/2020	7	28
71	Bolzano	3/13/2020	21	125
72	Lecce	3/13/2020	75	19
73	Ragusa	3/13/2020	88	2
74	Aosta	3/14/2020	7	42
75	Bolzano	3/14/2020	21	173
76	Lecce	3/14/2020	75	24
77	Ragusa	3/14/2020	88	2

11. Si calcola il numero di nuovi positivi a Marzo per la regioni coi mezzi di trasporto più sviluppati, ad esempio per quelle regioni con 2 o più aeroporti oppure con un numero di strade maggiore di 30.

```
SELECT R.codice_regione, R.denominazione_regione, R.numero_aereoporti,
R.numero_autostrade, (R.numero_autostrade+R.numero_strade_statali) AS Numero_strade,
SUM(C.nuovi_positivi) AS Persone_ammalatesi_a_Marzo
FROM COVID_REGIONI C JOIN REGIONI R ON C.codice_regione=R.codice_regione
WHERE data BETWEEN '1-Mar-2020' AND '31-Mar-2020' AND R.numero_aereoporti>=2
AND R.numero_autostrade>=1 AND (R.numero_autostrade + R.numero_strade_statali)>=30
GROUP BY R.codice_regione, R.denominazione_regione, R.numero_aereoporti,
R.numero_stazioni, R.numero_autostrade, R.numero_strade_statali
ORDER BY R.codice_regione;
```

1	CODICE_R	DENOMIN	NUMERO_AEREOI	NUMERO_AUTOSTR	NUMERO_STRADE	PERSONE_AMM
2	12	Lazio	12	2	37	3089
3	15	Campania	5	2	43	2079
4	18	Calabria	5	2	51	658
5	19	Sicilia	8	9	80	1643

12. A partire dal 1 Aprile, si selezionano le regioni e le date in cui le persone in terapia intensiva erano meno dei posti letto a disposizione, ovvero per quei giorni in cui la regione non ha dovuto richiedere posti aggiuntivi.

```
SELECT R.codice_regione, R.denominazione_regione, C.data,
R.posti_letto_terapia_intensiva, C.terapia_intensiva AS Persone_in_terapia_intensiva
FROM REGIONI R, COVID_REGIONI C
WHERE R.codice_regione=C.codice_regione AND (R.posti_letto_terapia_intensiva)>=
(C.terapia_intensiva) AND C.data>='01-Apr-2020'
ORDER BY C.data,R.codice_regione;
```

1	CODICE_R	DENOMINAZIONE	DATA	POSTI_LET	PERSONE_
2	1	Piemonte	4/1/2020	827	453
3	2	Valle d'Aosta	4/1/2020	35	27
4	5	Veneto	4/1/2020	825	350
5	6	Friuli Venezia Giu	4/1/2020	213	60
6	7	Liguria	4/1/2020	374	179
7	8	Emilia-Romagna	4/1/2020	708	359
8	9	Toscana	4/1/2020	569	297
9	10	Umbria	4/1/2020	105	45
10	11	Marche	4/1/2020	217	168
11	12	Lazio	4/1/2020	808	177
12	13	Abruzzo	4/1/2020	172	71
13	14	Molise	4/1/2020	34	8
14	15	Campania	4/1/2020	440	129
15	16	Puglia	4/1/2020	531	107
16	17	Basilicata	4/1/2020	73	15
17	18	Calabria	4/1/2020	206	16
18	19	Sicilia	4/1/2020	730	72
19	20	Sardegna	4/1/2020	158	27
20	98	P.A. Bolzano	4/1/2020	86	57
21	99	P.A. Trento	4/1/2020	80	76

13. Si seleziona la regione con il maggior numero di tamponi fatti.

```
SELECT C.codice_regione, R.denominazione_regione
FROM COVID_REGIONI C JOIN REGIONI R ON C.codice_regione=R.codice_regione
WHERE C.tamponi >= ALL(
SELECT MAX(C1.tamponi)
FROM COVID_REGIONI C1
GROUP BY C1.codice_regione);
```

1	CODICE_REGIONE	DENOMINAZIONE_REGIONE
2	3	Lombardia

14. Si calcola, per ogni regione, il numero di posti letto in terapia intensiva per ogni 10000 abitanti.

```
SELECT R.codice_regione, R.denominazione_regione,
(10000*R.posti_letto_terapia_intensiva/R.numero_abitanti) AS Posti_letto_terapia_intensiva
FROM REGIONI R
ORDER BY R.codice_regione;
```

1	CODICE_	DENOMINAZIONE_R	POSTI_LETTO
2	1	Piemonte	1,89835
3	2	Valle d'Aosta	2,78516
4	3	Lombardia	1,29118
5	5	Veneto	1,68166
6	6	Friuli Venezia Giulia	1,75277
7	7	Liguria	2,41191
8	8	Emilia-Romagna	1,58763
9	9	Toscana	1,52562
10	10	Umbria	1,19046
11	11	Marche	1,4227
12	12	Lazio	1,37436
13	13	Abruzzo	1,3114
14	14	Molise	1,1125
15	15	Campania	0,7584
16	16	Puglia	1,31793
17	17	Basilicata	1,29693
18	18	Calabria	1,05797
19	19	Sicilia	1,46003
20	20	Sardegna	0,96365
21	98	P.A. Bolzano	1,61904
22	99	P.A. Trento	1,47848

15. Si seleziona la regione col minor numero di decessi.

```
SELECT C.codice_regione, R.denominazione_regione, C.deceduti
FROM COVID_REGIONI C JOIN REGIONI R ON C.codice_regione=R.codice_regione
WHERE C.data='03-Mag-2020' AND C.deceduti =(
SELECT MIN(C1.deceduti)
FROM COVID_REGIONI C1
WHERE C1.data=C.data);
```

1	CODICE_REGIONE	DENOMINAZIONE	DECEDUTI
2	14	Molise	22

16. Si selezionano le regioni e i giorni in cui il numero dei positivi attuali è diminuito rispetto al giorno precedente.

```
SELECT C.data, R.codice_regione, R.denominazione_regione, C.variazione_totale_positivi
FROM COVID_REGIONI C, REGIONI R
WHERE C.codice_regione=R.codice_regione AND C.variazione_totale_positivi<0
ORDER BY C.data;
```

1	DATA	CODICE_	DENOMINAZIONE_	VARIAZIONE_TOT
2	2/26/2020	12	Lazio	-2
3	2/27/2020	19	Sicilia	-1
4	2/27/2020	1	Piemonte	-1
5	3/1/2020	7	Liguria	-17
6	3/2/2020	19	Sicilia	-2
7	3/2/2020	7	Liguria	-3
8	3/9/2020	1	Piemonte	-18
9	3/10/2020	3	Lombardia	-63
10	3/16/2020	14	Molise	-2
11	3/23/2020	14	Molise	-2
12	3/25/2020	2	Valle d'Aosta	-4
13	3/25/2020	14	Molise	-2
14	3/29/2020	10	Umbria	-1
15	3/30/2020	2	Valle d'Aosta	-21
16	3/30/2020	10	Umbria	-63
17	3/30/2020	6	Friuli Venezia Giul	-32
18	3/30/2020	3	Lombardia	-386
19	4/1/2020	98	P.A. Bolzano	-30
20	4/1/2020	2	Valle d'Aosta	-12
21	4/4/2020	17	Basilicata	-3
22	4/4/2020	11	Marche	-134

17. Si calcola la percentuale dei tamponi risultati positivi rispetto ai tamponi totali, fino al 5 Aprile.

```
SELECT R.codice_regione, R.denominazione_regione, (C.casi_totali*100)/(C.tamponi) AS
Percentuale_tamponi_usciti_pos
FROM COVID_REGIONI C JOIN REGIONI R ON C.codice_regione=R.codice_regione
WHERE C.data='05-Apr-2020'
ORDER BY R.codice_regione, R.denominazione_regione;
```

1	CODICE_	DENOMINAZIONE_REG	PERCENTUALE_
2	1	Piemonte	32,0766
3	2	Valle d'Aosta	32,7197
4	3	Lombardia	33,6403
5	5	Veneto	7,96679
6	6	Friuli Venezia Giulia	9,45871
7	7	Liguria	29,5674
8	8	Emilia-Romagna	24,4177
9	9	Toscana	11,4634
10	10	Umbria	9,95341
11	11	Marche	29,2683
12	12	Lazio	8,68825
13	13	Abruzzo	12,5239
14	14	Molise	14,8936
15	15	Campania	12,7923
16	16	Puglia	11,5388
17	17	Basilicata	9,48482
18	18	Calabria	6,07938
19	19	Sicilia	9,10336
20	20	Sardegna	12,6729
21	98	P.A. Bolzano	10,456
22	99	P.A. Trento	21,8118

18. Si seleziona la media delle persone in isolamento domiciliare nelle isole, per il mese di Marzo.

```
SELECT C.codice_regione, R.denominazione_regione, AVG(C.isolamento_domiciliare) AS Risultato_query
FROM COVID_REGIONI C, REGIONI R
WHERE R.codice_regione=C.codice_regione AND C.data BETWEEN '01-Mar-2020' AND '31-Mar-2020'
GROUP BY C.codice_regione, R.denominazione_regione
HAVING (R.denominazione_regione='Sicilia' OR R.denominazione_regione='Sardegna')
ORDER BY C.codice_regione;
```

1	CODICE_	DENOMINAZIONE_	RISULTATO_QUERY
2	19	Sicilia	254,742
3	20	Sardegna	154,387

6. La specifica in PL/SQL di una serie di procedure, funzioni e trigger

Una stored procedure è un programma, che utilizza l'SQL, memorizzato sul DBMS, in modo tale che esso possa essere richiamato e utilizzato più e più volte.

Il PL/SQL è un linguaggio di programmazione attraverso cui utenti e applicazioni possono accedere ed elaborare le informazioni gestite da un database ORACLE. In questo linguaggio le classiche istruzioni SQL sono arricchite da altre istruzioni tipiche dei linguaggi procedurali, come ad esempio gli statement condizionali e i loop. I blocchi che compongono un programma scritto in PL/SQL possono essere di diversi tipi, quelli di interesse in questa sezione sono i sottoprogrammi, ovvero le procedure e le funzioni, e i trigger.

Le procedure e le funzioni non sono altro che blocchi di istruzioni in PL/SQL che è possibile attivare mediante una chiamata a partire da un programma principale, e, come studiato nei linguaggi procedurali, esse possono essere dotate di parametri in ingresso e/o in uscita. La differenza tra una procedura e una funzione sta nel fatto che la funzione restituisce sempre un unico valore in uscita. L'attivazione di una procedura o di una funzione nel programma principale avviene mediante l'istruzione EXEC.

Di seguito si riportano una serie di procedure e funzioni che risultano utili quando si interagisce con la base di dati finora definita. Per ogni procedura/funzione è anche riportata una possibile istruzione EXEC che la attiva.

1. Si scrive una procedura che, ricevendo in ingresso la denominazione di una regione, restituisca in uscita il codice della provincia, la sigla della provincia, la regione a cui appartiene e il codice della regione.

```
CREATE OR REPLACE PROCEDURE prov
(provincia IN PROVINCE.denominazione_provincia%TYPE
)
AS
BEGIN
DECLARE
cod_prov PROVINCE.codice_provincia%TYPE;
sigla_prov PROVINCE.sigla_provincia%TYPE;
cod_reg REGIONI.codice_regione%TYPE;
regione REGIONI.denominazione_regione%TYPE;
trentino_alto_adige EXCEPTION;
BEGIN
SELECT PROVINCE.sigla_provincia, PROVINCE.codice_provincia, REGIONI.codice_regione,
REGIONI.denominazione_regione into sigla_prov, cod_prov, cod_reg, regione
FROM REGIONI JOIN PROVINCE ON REGIONI.codice_regione=PROVINCE.codice_regione
AND PROVINCE.denominazione_provincia=provincia;
if (cod_reg=98 OR cod_reg=99) then raise trentino_alto_adige;
else
DBMS_OUTPUT.PUT_LINE ('Nome della provincia: ' || provincia || '. Sigla della provincia: ' ||
sigla_prov || '. Codice della provincia: ' || cod_prov || '. Nome della regione: ' || regione || '.
Codice della regione: ' || cod_reg || '.');
```

```
end if;
EXCEPTION
when trentino_alto_adige then DBMS_OUTPUT.PUT_LINE ('Nome della provincia: ' || provincia || '. Sigla della provincia: ' || sigla_prov || '. Codice della provincia: ' || cod_prov || '. La provincia non fa parte di alcuna regione.' );
END;
END;
```

```
EXEC prov('Modena');
```

2. Si crea una procedura che riceva in ingresso una data e la denominazione di una regione e restituisca in uscita il numero delle persone contagiate fino a quel giorno.

```
CREATE OR REPLACE PROCEDURE conta_contagiati_per_regione
(regione IN REGIONI.denominazione_regione%TYPE,
 giorno IN COVID_REGIONI.data%TYPE
)
AS
BEGIN
DECLARE
persone_contagiate COVID_REGIONI.nuovi_positivi%TYPE;
cod_reg COVID_REGIONI.codice_regione%TYPE;
cont INTEGER;
regione_errata EXCEPTION;
BEGIN
SELECT DISTINCT COVID_REGIONI.codice_regione into cod_reg
FROM COVID_REGIONI JOIN REGIONI ON COVID_REGIONI.codice_regione =
REGIONI.codice_regione
WHERE REGIONI.denominazione_regione=regione;
SELECT COUNT (*) into cont FROM COVID_REGIONI WHERE codice_regione=cod_reg;
if (cont=0) then raise regione_errata;
else
SELECT SUM (nuovi_positivi) AS numero_contagiati into persone_contagiate
FROM COVID_REGIONI WHERE codice_regione=cod_reg AND data<=giorno
GROUP BY codice_regione;
DBMS_OUTPUT.PUT_LINE ('Numero di contagiati in ' || regione || ' fino al giorno ' || giorno || ':
' || persone_contagiate);
end if;
EXCEPTION
when regione_errata then DBMS_OUTPUT.PUT_LINE ('La regione selezionata non esiste.');
END;
END;
```

```
EXEC conta_contagiati_per_regione ('Campania','28-Mar-2020');
```

3. Si crea una procedura che, per un determinato giorno e per una data regione, controlli che il numero di persone ricoverate in terapia intensiva sia minore dei posti letto a disposizione, altrimenti stampa a schermo un allarme.

```
CREATE OR REPLACE PROCEDURE posti_letto_allarme
(giorno IN COVID_REGIONI.data%TYPE,
 regione IN REGIONI.denominazione_regione%TYPE
)
AS
BEGIN
DECLARE
max_posti_letto REGIONI.posti_letto_terapia_intensiva%TYPE;
persone_terapia_int COVID_REGIONI.terapia_intensiva%TYPE;
cod_reg REGIONI.codice_regione%TYPE;
BEGIN
SELECT codice_regione into cod_reg
FROM REGIONI WHERE denominazione_regione=regione;
SELECT terapia_intensiva into persone_terapia_int FROM COVID_REGIONI
WHERE data=giorno AND codice_regione=cod_reg;
SELECT posti_letto_terapia_intensiva into max_posti_letto FROM REGIONI
WHERE codice_regione=cod_reg;
if max_posti_letto<persone_terapia_int then DBMS_OUTPUT.PUT_LINE('Attenzione!');
end if;
END;
END;
```

```
EXEC posti_letto_allarme ('25-Apr-2020','Lombardia');
```

4. Si crea una proceduta che, dato un certo giorno, restituisca le regioni che hanno meno di 3000 positivi attuali.

```
CREATE OR REPLACE PROCEDURE meno_di_tremila
(giorno IN COVID_REGIONI.data%TYPE
)
AS
BEGIN
DECLARE
numero_contagiati COVID_REGIONI.totale_positivi%TYPE;
regione REGIONI.denominazione_regione%TYPE;
cont integer;
CURSOR cr IS SELECT R.denominazione_regione, C.totale_positivi
FROM COVID_REGIONI C JOIN REGIONI R ON R.codice_regione=C.codice_regione
WHERE C.data=giorno AND C.totale_Positivi<3000;
BEGIN
OPEN cr;
LOOP
FETCH cr into regione, numero_contagiati;
EXIT WHEN cr%NOTFOUND;
```

```
DBMS_OUTPUT.PUT_LINE ('Regione di ' || REGIONE || ', numero di contagiati: ' ||
numero_contagiati);
END LOOP;
CLOSE cr;
END;
END;
```

```
EXEC meno_di_tremila ('15-Apr-2020');
```

5. Si crea una procedura che, dato un intervallo di tempo, calcoli la regione che ha avuto il massimo numero di persone ospedalizzate.

```
CREATE OR REPLACE PROCEDURE ospedalizzati_procedure
( giorno1 IN COVID_REGIONI.data%TYPE,
  giorno2 IN COVID_REGIONI.data%TYPE
)
AS
BEGIN
DECLARE
ospedalizzati COVID_REGIONI.totale_ospedalizzati%TYPE;
regione COVID_REGIONI.codice_regione%TYPE;
giorno COVID_REGIONI.data%TYPE;
max_osp COVID_REGIONI.totale_ospedalizzati%TYPE :=0;
max_regione COVID_REGIONI.codice_regione%TYPE;
max_giorno COVID_REGIONI.data%TYPE;

errore EXCEPTION;

CURSOR cursore IS SELECT data, codice_regione, totale_ospedalizzati

FROM COVID_REGIONI WHERE data>=giorno1 AND data<=giorno2;
BEGIN
if giorno1<=giorno2 then
OPEN cursore;
LOOP
FETCH cursore into giorno, regione, ospedalizzati;
if ospedalizzati>max_osp then
max_giorno:=giorno; max_osp:=ospedalizzati; max_regione:=regione;
end if;
EXIT WHEN cursore%NOTFOUND;
END LOOP;
CLOSE cursore;
else raise errore;
end if;
DBMS_OUTPUT.PUT_LINE ('Il massimo degli ospedalizzati si ha per il giorno ' || max_giorno ||
' nella regione con codice = ' || max_regione || ', e il totale degli ospedalizzati è = ' || max_osp);
EXCEPTION
when errore then DBMS_OUTPUT.PUT_LINE ('I giorni non sono nell''ordine giusto.');
END;
END;
```

```
EXEC ospedalizzati_procedure ('13-Mar-2020','23-Mar-2020');
```

6. Si crea una procedura che, dato un certo giorno e una certa provincia, restituisca la differenza dei casi totali tra il giorno stesso e quello precedente.

```
CREATE OR REPLACE PROCEDURE differenza_tot_casi
(provincia IN PROVINCE.denominazione_provincia%TYPE,
 giorno_oggi IN COVID_PROVINCE.data%TYPE
)
AS
BEGIN
DECLARE
cod_prov COVID_PROVINCE.codice_provincia%TYPE;
differenza COVID_PROVINCE.totale_casi%TYPE;
tot_casi_oggi COVID_PROVINCE.totale_casi%TYPE;
tot_casi_ieri COVID_PROVINCE.totale_casi%TYPE;
giorno_ieri COVID_REGIONI.data%TYPE;
eccezione EXCEPTION;
BEGIN
SELECT codice_provincia into cod_prov FROM PROVINCE WHERE
denominazione_provincia=provincia;
SELECT DISTINCT data-1 into giorno_ieri FROM COVID_PROVINCE WHERE
data=giorno_oggi;
SELECT totale_casi into tot_casi_oggi FROM COVID_PROVINCE WHERE data=giorno_oggi
AND codice_provincia=cod_prov;
SELECT totale_casi into tot_casi_ieri FROM COVID_PROVINCE WHERE data=giorno_ieri
AND codice_provincia=cod_prov;
differenza := tot_casi_oggi - tot_casi_ieri;
if differenza <0 then raise eccezione;
else
DBMS_OUTPUT.PUT_LINE ('Totale dei casi nella provincia ' || provincia || ' nel giorno ' ||
giorno_ieri || ' = ' || tot_casi_ieri);
DBMS_OUTPUT.PUT_LINE ('Totale dei casi nella provincia ' || provincia || ' nel giorno ' ||
giorno_oggi || ' = ' || tot_casi_oggi);
DBMS_OUTPUT.PUT_LINE ('Differenza : ' || differenza);
end if;
EXCEPTION
when eccezione then DBMS_OUTPUT.PUT_LINE ('Attenzione! Vi è stato un errore
nell''inserimento dei dati');
END;
END;
```

```
EXEC differenza_tot_casi('Bergamo','14-Mar-2020');
```

7. Si crea una funzione la quale, dato un certo giorno, restituisca la denominazione della regione con il numero minore di persone in isolamento domiciliare.

```
CREATE OR REPLACE FUNCTION minor_num_isolamento
(giorno IN COVID_REGIONI.data%TYPE
) RETURN VARCHAR2
IS
BEGIN
DECLARE
isolamento COVID_REGIONI.isolamento_domiciliare%TYPE;
cod_reg COVID_REGIONI.codice_regione%TYPE;
min_isolamento COVID_REGIONI.isolamento_domiciliare%TYPE :=60360000;
min_cod_reg COVID_REGIONI.codice_regione%TYPE;
regione REGIONI.denominazione_regione%TYPE;
CURSOR cursore IS SELECT codice_regione, isolamento_domiciliare
FROM COVID_REGIONI WHERE data=giorno;
BEGIN
OPEN cursore;
LOOP
FETCH cursore into cod_reg, isolamento;
if isolamento < min_isolamento then
min_cod_reg := cod_reg; min_isolamento:=isolamento;
end if;
EXIT WHEN cursore%NOTFOUND;
END LOOP;
CLOSE cursore;
SELECT denominazione_regione INTO regione FROM REGIONI WHERE
codice_regione=min_cod_reg;
RETURN regione;
END;
END;
```

8. Si crea una funzione che riceve in ingresso il nome di una regione e una data e restituisce in uscita la percentuale dell'aumento dei casi totali rispetto al giorno precedente.

```
CREATE OR REPLACE FUNCTION percentuale_aumento_casi_tot
(regione IN REGIONI.denominazione_regione%TYPE,
 giorno_oggi IN COVID_REGIONI.data%TYPE
) RETURN NUMBER
IS
BEGIN
DECLARE
giorno_ieri COVID_REGIONI.data%TYPE;
cod_reg COVID_REGIONI.codice_regione%TYPE;
casi_oggi COVID_REGIONI.casi_totali%TYPE;
casi_ieri COVID_REGIONI.casi_totali%TYPE;
percentuale number;
BEGIN
```

```
SELECT codice_regione into cod_reg FROM REGIONI WHERE
denominazione_regione=regione;
SELECT DISTINCT data-1 into giorno_ieri FROM COVID_REGIONI WHERE
data=giorno_oggi;
SELECT casi_totali into casi_oggi FROM COVID_REGIONI WHERE data=giorno_oggi AND
codice_regione=cod_reg;
SELECT casi_totali into casi_ieri FROM COVID_REGIONI WHERE data=giorno_ieri AND
codice_regione=cod_reg;
percentuale := 100*(casi_oggi - casi_ieri) / casi_ieri;
RETURN percentuale;
END;
END;
```

Un trigger, invece, è una stored procedure che viene eseguita in maniera automatica a valle di particolari condizioni. Quando si definisce un trigger bisogna essere in possesso di tre informazioni: l'evento, che ha il compito di "accendere" il trigger, la condizione che deve essere verificata affinché il trigger acceso sia eseguito (se non specificata è sottinteso sempre TRUE), ed infine l'azione, ovvero una serie di istruzioni che devono essere eseguite se la condizione sul trigger acceso è soddisfatta.

Un trigger può attivarsi in seguito a un evento DML, di Data Manipulation Language, ovvero quando si ha a che fare con un INSERT, un UPDATE oppure un DELETE. Altrimenti un trigger può essere attivato a causa di statement DDL, di Data Definition Language, ovvero nel caso in cui l'evento attivante sia un CREATE, un ALTER o un DROP; oppure, più in generale, quando si attiva un evento di sistema, quale ad esempio l'avvio o la chiusura di un database.

Di seguito sono riportati alcuni esempi di trigger che si possono implementare sulla base di dati in modo da migliorarne le prestazioni. Inoltre, sono descritti, per ognuno di essi, dei possibili eventi attivanti.

1. Si crea un trigger che in seguito alla modifica del codice di una regione in REGIONI riporti la modifica anche nella tabella PROVINCE e COVID_REGIONI.

```
CREATE OR REPLACE TRIGGER modifica_cod_reg
AFTER UPDATE
OF codice_regione ON REGIONI
FOR EACH ROW
BEGIN
UPDATE PROVINCE SET codice_regione = :new.codice_regione WHERE
codice_regione=:old.codice_regione;
UPDATE COVID_REGIONI SET codice_regione = :new.codice_regione WHERE
codice_regione=:old.codice_regione;
END;
```

```
UPDATE REGIONI SET codice_regione=97 WHERE codice_regione=98;
```

2. Si crea un trigger che, in seguito a un INSERT nella tabella COVID, inserisca automaticamente le informazioni riguardanti le province nella tabella PROVINCE.

```
CREATE OR REPLACE TRIGGER inserimento_province
AFTER INSERT
ON COVID
FOR EACH ROW
BEGIN
INSERT INTO PROVINCE (codice_provincia, denominazione_provincia, sigla_provincia,
latitudine, longitudine) VALUES (:new.codice_provincia, :new.denominazione_provincia,
:new.sigla_provincia, :new.latitudine, :new.longitudine);
END;
```

```
INSERT INTO COVID VALUES ('27-Ago-2020', 'ITA', 8, 'Emilia-Romagna', 034, 'Parma', 'PR',
44.80107394, 10.32834985, 3886, NULL, NULL);
```

3. Si crea un trigger che, in seguito a un INSERT sulla tabella COVID, inserisca automaticamente le informazioni riguardanti le regioni nella tabella REGIONI.

```
CREATE OR REPLACE TRIGGER inserimento_regioni
AFTER INSERT
ON COVID
FOR EACH ROW
BEGIN
INSERT INTO REGIONI (stato, codice_regione, denominazione_regione) VALUES
(:new.stato, :new.codice_regione, :new.denominazione_regione);
END;
```

```
INSERT INTO COVID VALUES ('27-Ago-2020', 'ITA', 12, 'Lazio', 056, 'Viterbo', 'VT',
42.4173828, 12.10473416, 506, NULL, NULL);
```

4. Si crea un trigger che dopo un inserimento in COVID_REGIONI controlli che il numero dei positivi attuali sia minore del numero di abitanti della regione per cui si è fatto l'INSERT, altrimenti stampa a schermo un allarme.

```
CREATE OR REPLACE TRIGGER controllo_abitanti
AFTER INSERT
ON COVID_REGIONI
FOR EACH ROW
DECLARE
abitanti REGIONI.numero_abitanti%TYPE;
positivi_attuali COVID_REGIONI.totale_positivi%TYPE;
BEGIN
positivi_attuali := :new.totale_positivi;
SELECT numero_abitanti into abitanti FROM regioni WHERE
codice_regione=:new.codice_regione;
if positivi_attuali>abitanti then DBMS_OUTPUT.PUT_LINE('ALLARME');
end if; END;
```

```
INSERT INTO COVID_REGIONI (data, codice_regione, totale_positivi) VALUES ('13-Ago-2020', 15, 100000000000000);
```

5. Si crea un trigger che, in seguito alla cancellazione del capoluogo di una regione da PROVINCE, cancelli da REGIONI tutte le informazioni che riguardano la regione di cui la città è capoluogo.

```
CREATE OR REPLACE TRIGGER cancella_capoluogo
AFTER DELETE
ON PROVINCE
FOR EACH ROW
BEGIN
if (:old.denominazione_provincia='Aosta' OR :old.denominazione_provincia='Torino' OR
:old.denominazione_provincia='Genova' OR :old.denominazione_provincia='Milano' OR
:old.denominazione_provincia='Trento' OR :old.denominazione_provincia='Bolzano' OR
:old.denominazione_provincia='Venezia' OR :old.denominazione_provincia='Trieste' OR
:old.denominazione_provincia='Bologna' OR :old.denominazione_provincia='Firenze' OR
:old.denominazione_provincia='Perugia' OR :old.denominazione_provincia='Ancona' OR
:old.denominazione_provincia='Roma' OR :old.denominazione_provincia='L'Aquila' OR
:old.denominazione_provincia='Campobasso' OR :old.denominazione_provincia='Napoli' OR
:old.denominazione_provincia='Bari' OR :old.denominazione_provincia='Potenza' OR
:old.denominazione_provincia='Catanzaro' OR :old.denominazione_provincia='Palermo' OR
:old.denominazione_provincia='Cagliari')
then UPDATE REGIONI SET denominazione_regione=NULL,numero_abitanti=NULL,
superficie=NULL, densita=NULL, posti_letto_terapia_intensiva=NULL,
numero_aereoporti=NULL, numero_stazioni=NULL, numero_autostrade=NULL,
numero_strade_statali=NULL WHERE codice_regione= :old.codice_regione; end if;
END;
```

```
DELETE FROM PROVINCE WHERE denominazione_provincia='Potenza';
```

6. Si immagini l'esistenza di una tabella BACKUP_COVID_REGIONI, che presenta lo stesso schema relazionale di COVID_REGIONI. Si crea quindi un trigger che per ogni INSERT in COVID_REGIONI faccia l'inserimento che in BACKUP_COVID_REGIONI.

```
CREATE OR REPLACE TRIGGER backup_di_COVID_REGIONI
AFTER INSERT
ON COVID_REGIONI
FOR EACH ROW
BEGIN
INSERT INTO BACKUP_COVID_REGIONI VALUES (:new.data, :new.codice_regione,
:new.ricoverati_con_sintomi, :new.terapia_intensiva, :new.totale_ospedalizzati,
:new.isolamento_domiciliare, :new.variazione_totale_positivi, :new.totale_positivi,
:new.casi_totali, :new.nuovi_positivi, :new.dimessi_guariti, :new.deceduti, :new.tamponi,
:new.casi_testati);
END;
```

7. Come per il caso precedente, si crea un trigger che per ogni cancellazione fatta sulla tabella COVID_REGIONI faccia il DELETE anche per BACKUP_COVID_REGONI.

```
CREATE OR REPLACE TRIGGER backup_di_COVID_REGIONI
AFTER DELETE
ON COVID_REGIONI
FOR EACH ROW
BEGIN
DELETE FROM BACKUP_COVID_REGIONI WHERE data=:old.data AND codice_regione=
:old.codice_regione)
END;
```

7. La definizione di indici e viste sui dati memorizzati

Una vista è una tabella che viene descritta in termini di altre tabelle oppure, ricorsivamente, in termini di viste precedentemente descritte. La vista non contiene effettivamente i dati ma li riceve dalle tabelle su cui è basata. Una volta creata, la vista può essere trattata esattamente come una tabella, interrogandola.

Normalmente le tuple di una vista non sono memorizzate nella base di dati, quindi ogni qual volta deve essere utilizzata bisogna ricalcolare la query SQL che la definisce. Per evitare ciò, dato che spesso si può avere a che fare con query molto complesse e grandi moli di dati, la vista può essere "materializzata", ovvero si crea una tabella fisica che la rappresenta nel momento in cui viene fatta la query per la prima volta. In questo caso è però necessario definire delle politiche di aggiornamento automatico delle tabelle da cui la vista discende, altrimenti, se su di essa vengono fatte delle modifiche o degli inserimenti, si può avere a che fare con dati inconsistenti. Nel caso di viste non materializzate questo problema non sorge.

Di seguito sono riportate alcune viste che risultano utili quando si interagisce con la base di dati.

1. Si crea una vista che descriva l'andamento del contagio da COVID-19 per tutta Italia.

```
CREATE VIEW COVID_ITALIA AS
SELECT data AS Data, SUM(ricoverati_con_sintomi) AS Ricoverati_con_sintomi_it,
SUM(terapia_intensiva) AS Terapia_intensiva_it, SUM(totale_ospedalizzati) AS
Totale_ospedalizzati_it, SUM(isolamento_domiciliare) AS Isolamento_domiciliare_it,
SUM(variazione_totale_positivi) AS Variazione_totale_positivi_it, SUM(totale_positivi) AS
Totale_positivi_it, SUM(casi_totali) AS Casi_totali_it, SUM(nuovi_positivi) AS Nuovi_positivi_it,
SUM(dimessi_guariti) AS Dimessi_guariti_it, SUM(deceduti) AS Deceduti_it, SUM(tamponi)
AS Tamponi_it, SUM(casi_testati) AS Casi_testati_it
FROM COVID_REGIONI
GROUP BY data
ORDER BY data;
```

2. Si crea una vista materializzata con tutte le informazioni riguardo le regioni, prelevandole, eventualmente, anche dalla tabella PROVINCE.

```
CREATE MATERIALIZED VIEW REGIONI_TOT AS
SELECT R.codice_regione, R.denominazione_regione, R.numero_aereoporti,
R.numero_stazioni, R.numero_autostrade, R.numero_strade_statali,
COUNT(P.codice_provincia) AS Province_totali,  SUM(P.numero_comuni) AS Comuni_totali,
SUM (P.numero_scuole) AS Numero_scuole, SUM (P.numero_ospedali) AS
Numero_ospedali
FROM PROVINCE P JOIN REGIONI R ON R.codice_regione=P.codice_regione
GROUP BY R.codice_regione, R.denominazione_regione, R.numero_aereoporti,
R.numero_stazioni, R.numero_autostrade, R.numero_strade_statali
ORDER BY R.codice_regione;
```

3. Si crea una vista materializzata che "unisca" tutte le informazioni che riguardano le province di Trento e Bolzano, in modo da poter lavorare sulla regione Trentino-Alto Adige.

```
CREATE MATERIALIZED VIEW TRENTINO_ALTO_ADIGE AS
SELECT SUM(P.numero_abitanti) AS Numero_abitanti_TAA, SUM(P.numero_comuni) AS
Numero_comuni_TAA, SUM(P.superficie) AS Superficie_TAA, AVG(P.densita) AS
Densita_TAA, SUM(P.numero_scuole) AS Numero_scuole_TAA, SUM(P.numero_ospedali)
AS Numero_ospedali_TAA, SUM(R.posti_letto_terapia_intensiva) AS Posti_letto_TAA,
SUM(R.numero_aereoporti) AS Numero_aeroporti_TAA, SUM(R.numero_stazioni) AS
Numero_stazioni_TAA, SUM(R.numero_autostrade) AS Numero_autostrade_TAA,
SUM(R.numero_strade_statali) AS Numero_strade_statali_TAA
FROM PROVINCE P JOIN REGIONI R ON R.codice_regione=P.codice_regione
WHERE R.codice_regione=99 OR R.codice_regione=98;
```

4. Si crea una vista materializzata che rappresenti la realtà delle infrastrutture italiane, a partire dalle informazioni presenti nelle relazioni PROVINCE e REGIONI.

```
CREATE MATERIALIZED VIEW ITALIA AS
SELECT SUM(P.numero_abitanti) AS Numero_abitanti_it, SUM(P.numero_comuni) AS
Numero_comuni_it, SUM(P.superficie) AS Superficie_it, AVG(P.densita) AS Densita_it,
SUM(P.numero_scuole) AS Numero_scuole_it, SUM(P.numero_ospedali) AS
Numero_ospedali_it, SUM(R.posti_letto_terapia_intensiva) AS Posti_letto_it,
SUM(R.numero_aereoporti) AS Numero_aeroporti_it, SUM(R.numero_stazioni) AS
Numero_stazioni_it, SUM(R.numero_autostrade) AS Numero_autostrade_it,
SUM(R.numero_strade_statali) AS Numero_strade_statali_it
FROM PROVINCE P, REGIONI R
WHERE R.codice_regione=P.codice_regione;
```

5. Si crea una vista che descriva l'andamento del contagio da COVID-19 per tutto il Nord Italia.

```
CREATE VIEW COVID_NORD AS
SELECT data AS Data, SUM(ricoverati_con_sintomi) AS Ricoverati_sintomi_nord,
SUM(terapia_intensiva) AS Terapia_intensiva_nord, SUM(totale_ospedalizzati) AS
Totale_ospedal_nord, SUM(isolamento_domiciliare) AS Isolamento_domicil_nord,
SUM(variazione_totale_positivi) AS Variazione_tot_pos_nord, SUM(totale_positivi) AS
Totale_positivi_nord, SUM(casi_totali) AS Casi_totali_nord, SUM(nuovi_positivi) AS
Nuovi_positivi_nord, SUM(dimessi_guariti) AS Dimessi_guariti_nord, SUM(deceduti) AS
Deceduti_nord, SUM(tamponi) AS Tamponi_nord, SUM(casi_testati) AS Casi_testati_nord
FROM COVID_REGIONI
WHERE codice_regione=8 OR codice_regione=6 OR codice_regione=7 OR
codice_regione=3 OR codice_regione=1 OR codice_regione=99 OR codice_regione=98 OR
codice_regione=2 OR codice_regione=5
GROUP BY data
ORDER BY data;
```

6. Si crea una vista materializzata che contenga tutte le informazioni dei capoluoghi delle regioni, estraendole dalla tabella PROVINCE.

```
CREATE MATERIALIZED VIEW CAPOLUOGHI AS
SELECT R.denominazione_regione, P1.*
FROM PROVINCE P1 JOIN REGIONI R ON R.codice_regione=P.codice_regione
WHERE P1.denominazione_provincia IN (
SELECT P.denominazione_provincia FROM PROVINCE P WHERE
P.denominazione_provincia='Aosta' OR P.denominazione_provincia='Torino' OR
P.denominazione_provincia='Genova' OR P.denominazione_provincia='Milano' OR
P.denominazione_provincia='Trento' OR P.denominazione_provincia='Bolzano' OR
P.denominazione_provincia='Venezia' OR P.denominazione_provincia='Trieste' OR
P.denominazione_provincia='Bologna' OR P.denominazione_provincia='Firenze' OR
P.denominazione_provincia='Perugia' OR P.denominazione_provincia='Ancona' OR
P.denominazione_provincia='Roma' OR P.denominazione_provincia='L'Aquila' OR
P.denominazione_provincia='Campobasso' OR P.denominazione_provincia='Napoli' OR
P.denominazione_provincia='Bari' OR P.denominazione_provincia='Potenza' OR
P.denominazione_provincia='Catanzaro' OR P.denominazione_provincia='Palermo' OR
P.denominazione_provincia='Cagliari')
ORDER BY P1.codice_provincia;
```

7. Si crea una vista materializzata con le informazioni riguardo le regioni a statuto speciale.

```
CREATE MATERIALIZED VIEW STATUTO_SPECIALE AS
SELECT R.codice_regione, R.denominazione_regione, R.numero_aereoporti,
R.numero_stazioni, R.numero_autostrade, R.numero_strade_statali,
COUNT(P.codice_provincia) AS Province_totali,  SUM(P.numero_comuni) AS Comuni_totali,
SUM (P.numero_scuole) AS Numero_scuole, SUM (P.numero_ospedali) AS
Numero_ospedali
FROM PROVINCE P JOIN REGIONI R ON R.codice_regione=P.codice_regione
WHERE R.denominazione_regione='Sicilia' OR R.denominazione_regione='Sardegna' OR
R.denominazione_regione='Valle d''Aosta' OR R.denominazione_regione='Friuli Venezia
Giulia' OR R.denominazione_regione='P.A. Bolzano' OR R.denominazione_regione='P.A.
Trento'
GROUP BY R.codice_regione, R.denominazione_regione, R.numero_aereoporti,
R.numero_stazioni, R.numero_autostrade, R.numero_strade_statali
ORDER BY R.codice_regione;
```

Un indice è una tabella che contiene i valori della colonna o delle colonne di una relazione, e per ogni valore memorizza le posizioni delle tuple che lo contengono, proprio come se fosse l'indice di un libro. Durante lo svolgimento di una query, attraverso gli indici è possibile identificare, in maniera rapida, la posizione esatta dei dati su cui la query deve lavorare, evitando la scansione totale delle tabelle.

Di seguito sono riportato alcuni indici che si possono definire sulla base di dati.

```
CREATE INDEX idx_covid_reg ON COVID_REGIONI (data, codice_regione);
CREATE INDEX idx_covid_prov ON COVID_PROVINCE (data, codice_provincia);
CREATE INDEX idx_regione ON REGIONI (codice_regione,denominazione_regione);
CREATE INDEX idx_prov ON PROVINCE (codice_provincia,denominazione_provincia,
sigla_provincia);
```

Si noti che alcuni degli indici sono definiti proprio sulla chiave primaria della relazione, dato che questi attributi vengono spesso utilizzati nelle query.