Αναφορά Project B

Φίλιππος Δουραχαλής, 3170045

Ζήτημα 1)

1. Δημιουργούμε τη βάση CAMPDW μέσω της εντολής:

```
CREATE DATABASE CAMPDW;
```

Στη συνέχεια δημιουργούμε τον πίνακα campdata που θα περιέχει όλες τις εγγραφές του txt αρχείου που δίνεται και τον συμπληρώνουμε με τα δεδομένα του αρχείου χρησιμοποιώντας τις παρακάτω εντολές:

```
CREATE TABLE campdata(
	custID int not null,
	fname varchar(30) not null,
	lname varchar(30) not null,
	cID int not null,
	country varchar(30) not null,
	bookID int not null,
	bookDate date not null,
	campCode char(3) not null,
	CampName varchar(50) not null,
	empno int not null,
	catCode char not null,
	category varchar(20) not null,
	unitCost numeric(4,2) not null,
	startDate date not null,
	overnights int not null,
	persons int not null
);

BULK INSERT campdata
FROM 'C:\Users\Philip\Downloads\CAMPDATA.TXT'
	WITH (FIRSTROW =2, FIELDTERMINATOR='|', ROWTERMINATOR = '\n');
```

2. Το λογικό star schema της βάσης θα περιέχει τους πίνακες διαστάσεων:
«Τουριστικά Γραφεία» (Agencies) που θα περιέχει τον κωδικό του γραφείου, το ονοματεπώνυμο του υπευθύνου και τον κωδικό της χώρας στην οποία βρίσκεται,
«Κατασκηνώσεις» (Camps), που περιέχει τον κωδικό και το όνομα των κατασκηνώσεων,
«Χώρες» (Countries), που θα περιλαμβάνει τον κωδικό και το όνομα κάθε χώρας,

«Θέσεις» (Seats), που περιέχει τον κωδικό κάθε θέσης, το όνομά της και το κόστος της και τέλος
«Χρόνος» (TimeInfo), που περιέχει πληροφορίες για τον χρόνο έναρξης μιας κράτησης (πλήρης ημ/νια, ημέρα, μήνας, έτος και ημέρα του χρόνου).

Οι παραπάνω πίνακες θα μας επιτρέψουν να κάνουμε ανάλυση των δεδομένων μέσω των διαστάσεων που αναφέρονται.

Ο πίνακας γεγονότων (fact table) θα είναι ο **«Κρατήσεις» (Bookings)**, ο οποίος θα περιέχει πληροφορίες για μια κράτηση (κωδικό πελάτη που έκανε την κράτηση, χρόνο κράτησης και έναρξης, κωδικό κατασκήνωσης, κωδικός και αριθμός θέσης, κωδικός χώρας, διανυκτερεύσεις, άτομα και συνολικό κόστος) και θα συνδέεται με κάθε dimension table.

Οι εντολές δημιουργίας των ανωτέρω πινάκων είναι οι εξής:

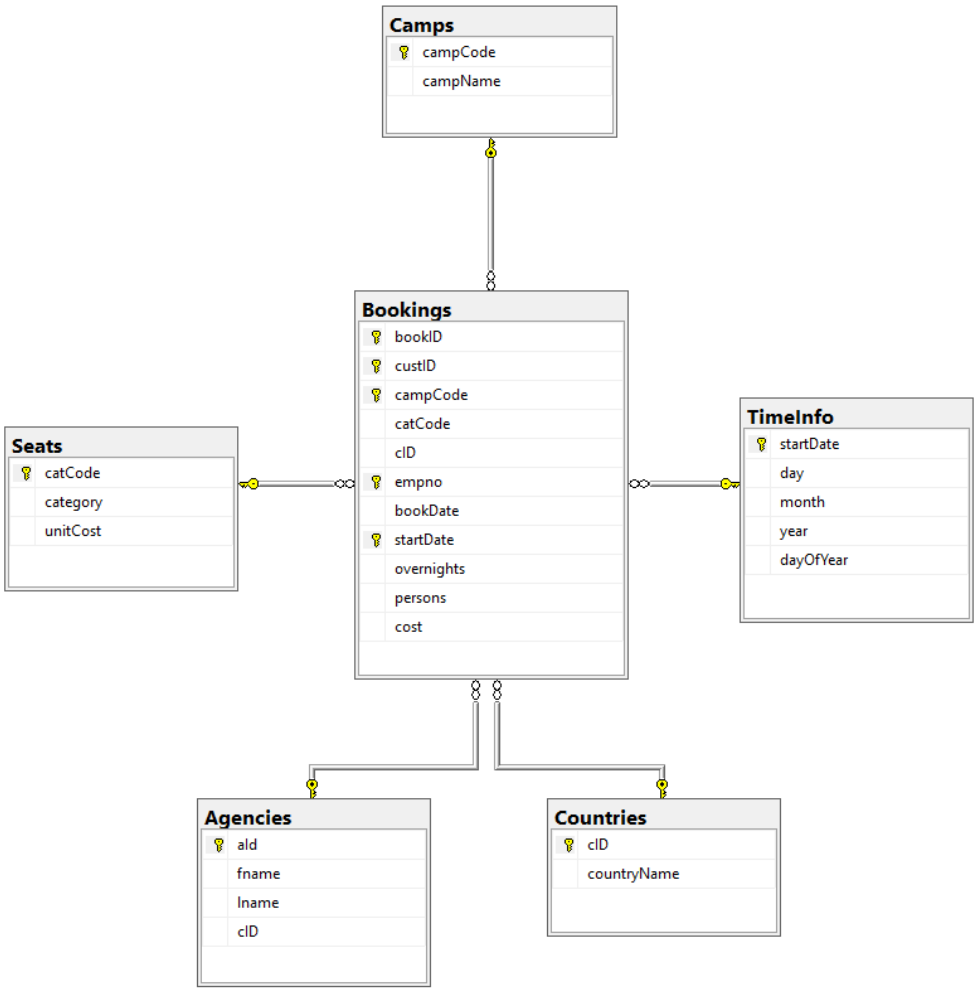
```
CREATE TABLE Countries(
	cID int primary key,
	countryName varchar(30) not null
);


CREATE TABLE Agencies (
	aId integer primary key,
	fname varchar(30) not null,
	lname varchar(30) not null,
	cID integer not null
);


CREATE TABLE Camps(
	campCode char(3) primary key,
	campName varchar(50) not null
);

CREATE TABLE Seats(
	catCode char(1) primary key,
	category varchar(20) not null,
	unitCost numeric(4,2) not null
);

CREATE TABLE TimeInfo(
	startDate date primary key,
	day int,
	month int,
	year int,
	dayOfYear int
);
```

```
CREATE TABLE Bookings(
	bookID int,
	custID int not null,
	campCode char(3) not null,
	catCode char(1) not null,
	cID int not null,
	empno int not null,
	bookDate date not null,
	startDate date not null,
	overnights int not null,
	persons int not null,
	cost numeric(6,2) not null

	primary key(bookID, custID, campCode, startDate, empno),
	foreign key(custID) references Agencies(aid),
	foreign key(campCode) references Camps(campCode),
	foreign key(catCode) references Seats(catCode),
	foreign key(startDate) references TimeInfo(startDate),
	foreign key(cID) references Countries(cID)
	);
```

3. Οι εντολές τροφοδοσίας των πινάκων είναι οι εξής:

```
INSERT INTO Agencies
	SELECT DISTINCT custId, fname, lname, cId
	FROM campdata
	ORDER BY custId;

INSERT INTO Countries
	SELECT DISTINCT cID, country
	FROM campdata
	ORDER BY cID;

INSERT INTO Camps
	SELECT DISTINCT campCode, campName
	FROM campdata
	ORDER BY campCode;

INSERT INTO Seats
	SELECT DISTINCT catCode, category, unitCost
	FROM campdata
	ORDER BY catCode;

set datefirst 1;
INSERT INTO TimeInfo
	SELECT DISTINCT startDate,
				DATEPART(day, startDate),
				DATEPART(month, startDate),
				DATEPART(year, startDate),
				DATEPART(dayofyear, startDate)
	FROM campdata
	ORDER BY startDate;

INSERT INTO Bookings
	SELECT DISTINCT bookID, custID, campCode, catCode, cID, empno,
			bookDate, startDate, overnights, persons,
			(overnights*persons*unitCost) as cost
	FROM campdata;
```

4. Στην παρακάτω εικόνα βλέπουμε το σχήμα της αποθήκης που παρήγαγε ο SQL Server:



Ζήτημα 2)

Παρατίθενται τα ερωτήματα SQL για κάθε υποερώτημα του συγκεκριμένου ζητήματος.

1.
```sql
SELECT countryName, fname, lname, total_profit
FROM  (SELECT TOP 100 custID, cID, sum(cost) as total_profit
            FROM Bookings
            GROUP BY custID, cID
            ORDER BY total_profit DESC) Q
INNER JOIN Agencies A ON A.aId = Q.custID
INNER JOIN Countries C ON C.cid = Q.cID
ORDER BY total_profit DESC;
```

Στο παραπάνω ερώτημα παίρνουμε τον κωδικό των 100 πρώτων σε κρατήσεις τουριστικών γραφείων μέσω του εμφωλευμένου ερωτήματος και στη συνέχεια κάνουμε JOIN τη σχέση που προκύπτει με τον πίνακα Agencies που αναπαριστά τα γραφεία, ώστε να βρούμε τις πληροφορίες του υπεύθυνου και με τον πίνακα Countries για να βρούμε το όνομα της χώρας του γραφείου.

2.
```
SELECT campName, category, profit
FROM (SELECT campCode, catCode, SUM(cost) as profit
      FROM Bookings B
      INNER JOIN TimeInfo T ON T.startDate = B.startDate
      WHERE T.year = 2000
      GROUP BY campCode, catCode ) Q, Camps C, Seats S
WHERE Q.campCode = C.campCode AND S.catCode = Q.catCode
ORDER BY C.campName, S.category
```

Σε αυτό το ερώτημα, βρίσκουμε τη συνολική αξία κρατήσεων ανά κωδικό κατασκήνωσης και θέσης το 2000 μέσα στο εμφωλευμένο ερώτημα και μετά παίρνουμε το καρτεσιανό γινόμενο της σχέσης που προκύπτει με τους πίνακες Camps και Seats για να βρούμε το όνομα της κατασκήνωση και την κατηγορία της κάθε θέσης αντίστοιχα.

3.
```
SELECT campName, month, profit
FROM (SELECT campCode, month, SUM(cost) as profit
             FROM Bookings B
             INNER JOIN TimeInfo T ON T.startDate = B.startDate
             WHERE T.year = 2018
             GROUP BY campCode, month ) Q
INNER JOIN Camps as C ON Q.campCode = C.campCode
ORDER BY campName, month
```

Αντίστοιχα με προηγουμένως, βρίσκουμε τις αξίες κρατήσεων ανα κωδικό κατασκήνωση και μήνα, φιλτράροντας μόνο τις εγγραφές για το έτος 2018 και στη συνέχεια μέσω του JOIN στο εξωτερικό ερώτημα βρίσκουμε τα ονόματα των κατασκηνώσεων.

4.
```
SELECT COALESCE(CONVERT(varchar(30), year), 'All Time') AS year,
       COALESCE(campCode, 'All camps') AS campCode,
COALESCE(catCode, 'All categories') AS catCode,
SUM(persons) as total_persons
FROM Bookings as B
INNER JOIN TimeInfo as T ON T.startDate = B.startDate
GROUP BY ROLLUP (year, campCode, catCode)
```

Στο συγκεκριμένο ερώτημα πρέπει να εφαρμόσουμε ουσιαστικά 4 διαφορετικά GROUP BY ώστε να βρούμε τα στοιχεία που ζητούνται. Προκειμένου να μην δημιουργηθούν 4 διαφορετικά ερωτήματα, χρησιμοποιείται στην πρόταση GROUP BY ο τελεστής ROLLUP, που μας επιτρέπει να υπολογίζουμε όλους τους ενοικιαστές για ένα ανεξάρτητο υποσύνολο των στηλών που εμφανίζονται στο group by (διαδοχική μείωση

διαστάσεων). Τα ξεχωριστά υποσύνολα που θα υπολογιστούν πρακτικά ταυτόχρονα μέσω του ROLLUP είναι τα εξής:

`GROUP BY (year, campCode, catCode):` Σύνολο ενοικιαστών ανά έτος, κατασκήνωση και θέση.

`GROUP BY (year, campCode):` Σύνολο ενοικιαστών ανά έτος και κατασκήνωση.

`GROUP BY (year):` Σύνολο ενοικιαστών ανά έτος

`GROUP BY none` Σύνολο ενοικιαστών συνολικά.

Τέλος εφαρμόζουμε τη συνάρτηση COALESCE για να δώσουμε πιο περιγραφικά ονόματα στα κελιά που θα περιέχουν αυτά τα υποσύνολα ώστε να μην εμφανίζεται null.

5. Για το τελευταίο ερώτημα δημιουργούμε αρχικά μια όψη που θα περιλαμβάνει τα στοιχεία των κρατήσεων ανά κατασκήνωση για το έτος 2018. Η δημιουργία της όψης γίνεται με την εντολή

```
CREATE VIEW PPC2018 AS
SELECT campCode, SUM(persons) as persons
FROM Bookings
INNER JOIN TimeInfo ON TimeInfo.startDate = Bookings.startDate
WHERE year = 2018
      GROUP BY campCode;
```

Αφού το κάνουμε αυτό υπολογίζουμε το ζητούμενο με το ερώτημα:

```
SELECT campName, P.persons
 FROM (SELECT campCode, SUM(persons) as persons
       FROM Bookings
       INNER JOIN TimeInfo ON TimeInfo.startDate = Bookings.startDate
       WHERE year = 2017
       GROUP BY campCode) Q
 INNER JOIN PPC2018 AS P ON P.campCode = Q.campCode
 INNER JOIN Camps C ON C.campCode = P.campCode
       WHERE P.persons > Q.persons
```

Σε αυτό το ερώτημα υπολογίζουμε το σύνολο κρατήσεων ανα κατασκήνωση για το 2017 (όπως το υπολογίσαμε και για την όψη). Στη συνέχεια κάνουμε JOIN τον πίνακα που προκύπτει με την όψη που φτιάξαμε προηγουμένως και ελέγχουμε αν για κάθε ίδια κατασκήνωση, το σύνολο ενοικιαστών που υπολογίσαμε στην όψη είναι μεγαλύτερο από το αντίστοιχο σύνολο που υπολογίσαμε στο εμφωλευμένο ερώτημα.

Ζήτημα 3)

Δημιουργούμε έναν κύβο με τις διαστάσεις «Μήνας», «Κατασκήνωση» και «Χώρα». Ο κύβος δημιουργείται με την εντολή:

```
SELECT campName, countryName, month, SUM(cost) as income
FROM Bookings AS B
INNER JOIN TimeInfo AS T ON T.startDate = B.startDate
INNER JOIN Camps AS C ON C.campCode = B.campCode
INNER JOIN Countries AS Cnt ON Cnt.cID = B.cID
GROUP BY CUBE (month, campName, countryName);
```

a. Κάθε κελί του κύβου θα περιέχει την αξία των κρατήσεων που πραγματοποιήθηκαν σε έναν ορισμένο μήνα από τουριστικά γραφεία σε συγκεκριμένη χώρα και για μια συγκεκριμένη κατασκήνωση. Πρακτικά από τον συγκεκριμένο κύβο μπορούμε να λάβουμε πληροφορίες όπως ποιοι μήνες παράγουν περισσότερα έσοδα και από ποιες χώρες γίνονται οι περισσότερες κρατήσεις και προς ποιες κατασκηνώσεις μεταξύ άλλων.

b. Μέσω του κύβου μπορούμε να υπολογίσουμε οποιοδήποτε συνδυασμό των 3 διαστάσεων που αναφέρθηκαν προηγουμένως. Γνωρίζουμε ότι όλοι οι πιθανοί συνδυασμοί (δηλαδή τα GROUP BY) είναι $2^3 = 8$ και θα είναι οι εξής:

   i) `GROUP BY none` (συνολική αξία ανεξαρτήτου χώρας, μήνα και κατασκήνωσης)

   ii) `GROUP BY month` (συνολική αξία ανά μήνα)

   iii) `GROUP BY campName` (συνολική αξία ανά κατασκήνωση)

   iv) `GROUP BY countryName` (συνολική αξία ανά χώρα)

   v) `GROUP BY (month, campName)` (συνολική αξία ανά μήνα και ανά κατασκήνωση)

   vi) `GROUP BY (month, countryName)` (συνολική αξία ανά μήνα και ανά χώρα)

   vii) `GROUP BY (campName, countryName)` (συνολική αξία ανά κατασκήνωση και χώρα)

   viii) `GROUP BY (month, campName, countryName)` (συνολική αξία ανά μήνα, κατασκήνωση και ανά χώρα)

Ζήτημα 4)

Ο πίνακας countries περιέχει 10 διαφορετικές εγγραφές, δηλαδή 10 χώρες.

Το bitmap ευρετήριο θα κωδικοποιεί για κάθε χώρα τις αντίστοιχες θέσεις των εγγραφών του fact table που περιέχουν τη χώρα αυτή. Η μορφή του ευρετηρίου θα είναι η ακόλουθη:

fact_countries_idx

row#	Germany	Italy	France	Spain	Sweden	Denmark	Netherlands	Austria	Belgium	Finland
69	0	0	0	1	0	0	0	0	0	0
70	0	0	0	1	0	0	0	0	0	0
71	0	0	0	0	0	1	0	0	0	0
72	0	0	0	0	0	0	0	1	0	0
73	0	0	0	0	0	0	0	1	0	0
74	1	0	0	0	0	0	0	0	0	0
75	1	0	0	0	0	0	0	0	0	0
76	1	0	0	0	0	0	0	0	0	0