ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

ΤΜΗΜΑ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Μάθημα: ***Βάσεις Δεδομένων***

Εργασία 2019-2020

Τεχνικό Εγχειρίδιο

Συμμετέχουν οι φοιτητές:

Όνομα: Παναγιώτης
Επώνυμο: Υφαντής
Α.Μ.: Ε16155

Όνομα: Αντώνιος
Επώνυμο: Παπαχρήστου
Α.Μ.: Ε14152

Στάδια εργασίας

Α) Δημιουργία εντολών SQL

Σε αυτό το στάδιο γίνεται η παρουσίαση των Stored Procedures που υλοποιήθηκαν γιατί εκτέλεση των ερωτημάτων των αναφορών.

1) Ανέκτησε τους καλλιτέχνες που οι δίσκοι τους είναι ανάμεσα στα πρώτα Χ σε πωλήσεις για συγκεκριμένο χρονικό διάστημα. Ο χρήστης θα πρέπει να ορίζει τα εξής κριτήρια για την δημιουργία της αναφοράς: Αριθμός Χ, Χρονικό διάστημα Ημερομηνία Από – Έως.

Παρακάτω παρουσιάζεται το Stored Procedure "spXTopArtists" που περιλαμβάνει τον κώδικα SQL για την υλοποίηση του ερωτήματος.

spXTopArtists.sql -...-JVJ5FIF\Panos (52))

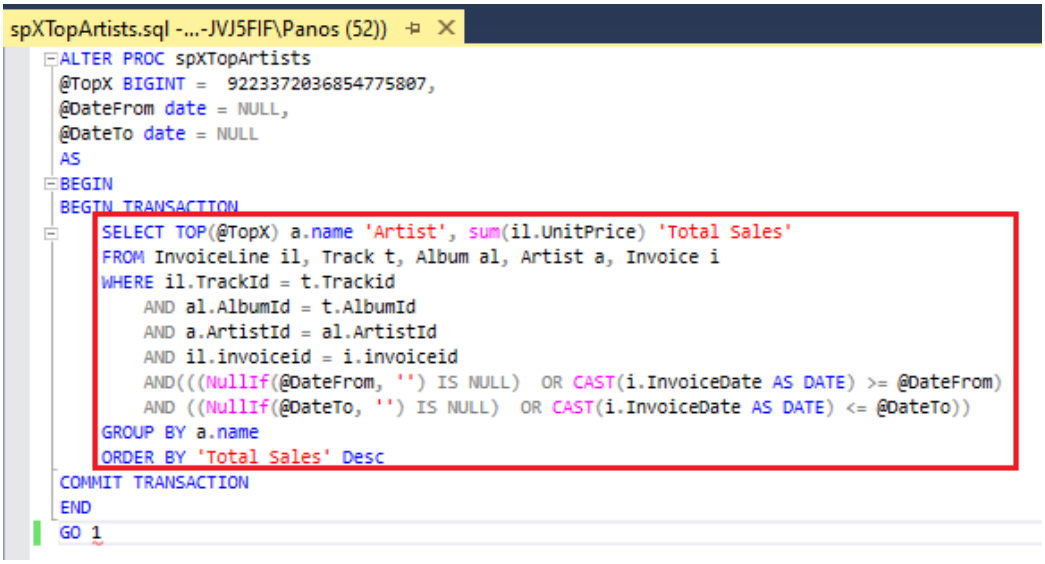
```
ALTER PROC spXTopArtists
@TopX BIGINT =  9223372036854775807,
@DateFrom date = NULL,
@DateTo date = NULL
AS
BEGIN
BEGIN TRANSACTION
    SELECT TOP(@TopX) a.name 'Artist', sum(il.UnitPrice) 'Total Sales'
    FROM InvoiceLine il, Track t, Album al, Artist a, Invoice i
    WHERE il.TrackId = t.Trackid
        AND al.AlbumId = t.AlbumId
        AND a.ArtistId = al.ArtistId
        AND il.invoiceid = i.invoiceid
        AND(((NullIf(@DateFrom, '') IS NULL)  OR CAST(i.InvoiceDate AS DATE) >= @DateFrom)
        AND ((NullIf(@DateTo, '') IS NULL)  OR CAST(i.InvoiceDate AS DATE) <= @DateTo))
    GROUP BY a.name
    ORDER BY 'Total Sales' Desc
COMMIT TRANSACTION
END
GO 1
```

Το παραπάνω Stored Procedure λαμβάνει τις παραμέτρους τον αριθμό των Top X καλλιτεχνών και δύο ημερομηνίες Από – Έως. Σε περίπτωση που δεν δοθούν τιμές στα πεδία δίνονται Default values.

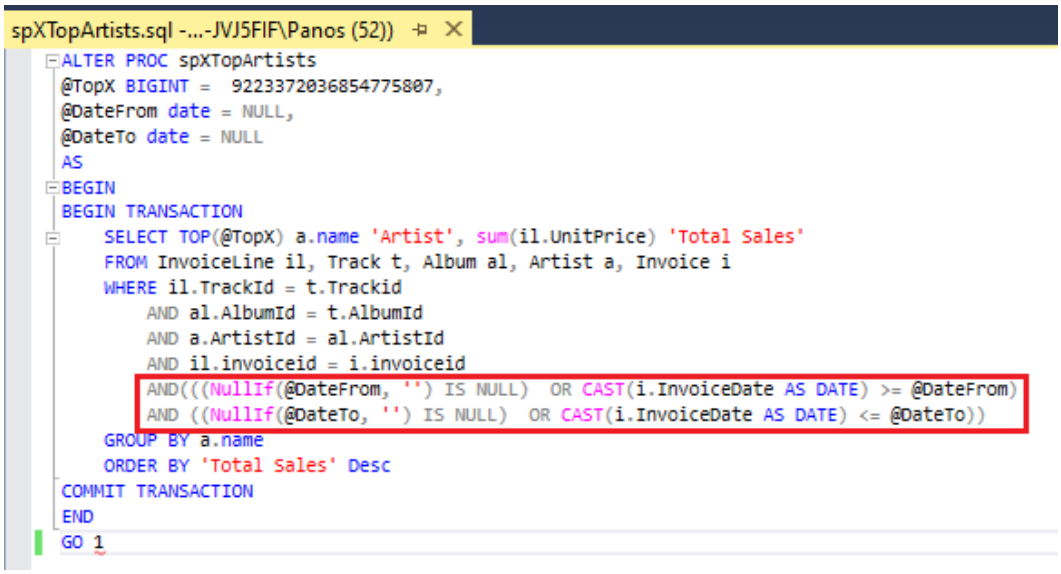
spXTopArtists.sql -...-JVJ5FIF\Panos (52))

```
ALTER PROC spXTopArtists
@TopX BIGINT =  9223372036854775807,
@DateFrom date = NULL,
@DateTo date = NULL
AS
BEGIN
BEGIN TRANSACTION
    SELECT TOP(@TopX) a.name 'Artist', sum(il.UnitPrice) 'Total Sales'
    FROM InvoiceLine il, Track t, Album al, Artist a, Invoice i
    WHERE il.TrackId = t.Trackid
        AND al.AlbumId = t.AlbumId
        AND a.ArtistId = al.ArtistId
        AND il.invoiceid = i.invoiceid
        AND(((NullIf(@DateFrom, '') IS NULL)  OR CAST(i.InvoiceDate AS DATE) >= @DateFrom)
        AND ((NullIf(@DateTo, '') IS NULL)  OR CAST(i.InvoiceDate AS DATE) <= @DateTo))
    GROUP BY a.name
    ORDER BY 'Total Sales' Desc
COMMIT TRANSACTION
END
GO 1
```

Στη συνέχεια γίνεται επιλογή των ονομάτων των Καλλιτεχνών καθώς και των συνολικών τους πωλήσεων. Αυτό επιτυγχάνεται λαμβάνοντας της εγγραφές εκείνες που ικανοποιούν τους ελέγχους των πρωτεύων και δευτερευόντων κλειδιών και τα αποτελέσματα παρουσιάζονται σε φθίνουσα σειρά ως προς των αριθμό των πωλήσεων.

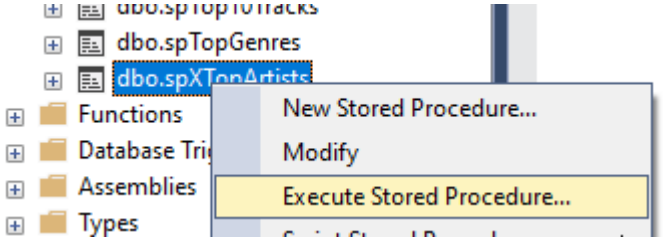


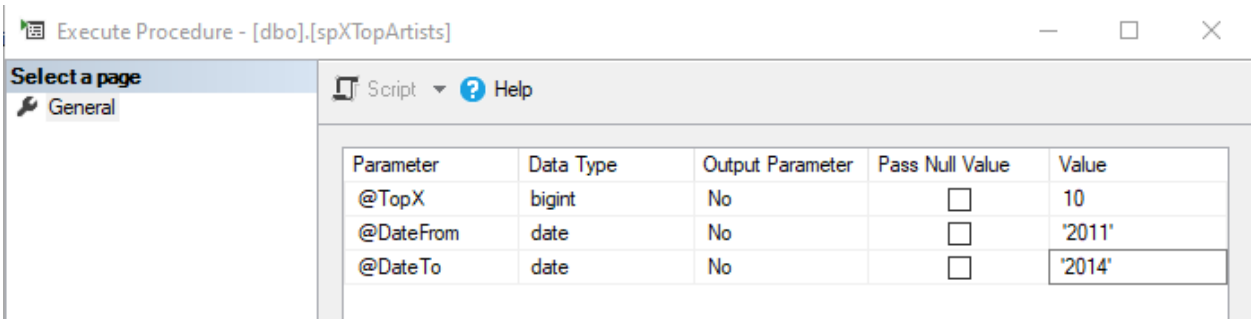
Συγκεκριμένα οι δύο τελευταίες εντολές επιτελούν το σκοπό της ανάθεσης της τιμής Null στο ανάλογο πεδίο όταν δεν ικανοποιείται οι συγκεκριμένη συνθήκη, δηλαδή ο χρήστης έχει αφήσει κενό πεδίο και σε αντίθετη περίπτωση παραθέτετε η δοθείσα τιμή και γίνεται ο έλεγχος του διαστήματος.



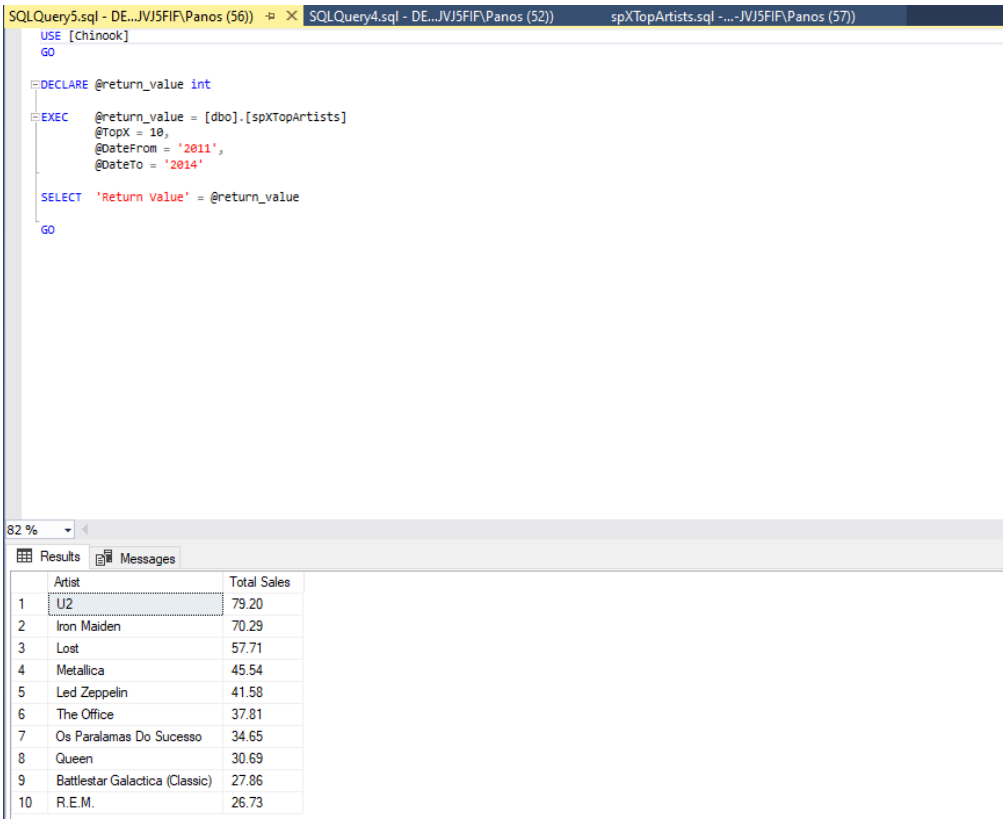
Προχωράμε λοιπόν στην εκτέλεση της παραπάνω διαδικασίας.

Επιλέγουμε λοιπόν το όνομα της συγκεκριμένης διαδικασίας και πατάμε execute, έπειτα συμπληρώνουμε τα πεδία που μας ενδιαφέρουν και πατάμε οκ. Στη προκειμένη περίπτωση θέλουμε να δούμε τους πρώτους 10 καλλιτέχνες σε πωλήσεις μεταξύ του 2011 και του 2014.





Και έχουμε τα ανάλογα αποτελέσματα



	Artist	Total Sales
1	U2	79.20
2	Iron Maiden	70.29
3	Lost	57.71
4	Metallica	45.54
5	Led Zeppelin	41.58
6	The Office	37.81
7	Os Paralamas Do Sucesso	34.65
8	Queen	30.69
9	Battlestar Galactica (Classic)	27.86
10	R.E.M.	26.73

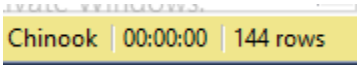
Επαναλαμβάνουμε την ίδια διαδικασία μόνο που στη προκειμένη περίπτωση δεν παραθέτουμε τον αριθμό των top καλλιτεχνών και μας ενδιαφέρουν οι πωλήσεις μόνο από το 2012 και μετά.

Parameter	Data Type	Output Parameter	Pass Null Value	Value
@TopX	bigint	No	☐	
@DateFrom	date	No	☐	'2012'
@DateTo	date	No	☐	

Και έχουμε τα εξής αποτελέσματα

	Artist	Total Sales
1	Iron Maiden	69.30
2	U2	52.47
3	Metallica	42.57
4	Led Zeppelin	38.61
5	Lost	35.82
6	The Office	27.86
7	Os Paralamas Do Sucesso	22.77
8	Queen	19.80
9	Smashing Pumpkins	15.84
10	Pearl Jam	14.85
11	R.E.M.	13.86
12	The Who	13.86
13	Lenny Kravitz	12.87
14	Red Hot Chili Peppers	11.88
15	Tim Maia	11.88
16	Titãs	11.88
17	Van Halen	11.88

130	Berliner Philharmoniker & ...	0.99
131	Apocalyptica	0.99
132	Alice In Chains	0.99
133	Antal Doráti & London Sy...	0.99
134	Academy of St. Martin in ...	0.99
135	Academy of St. Martin in ...	0.99
136	Accept	0.99
137	Aerosmith	0.99
138	Chicago Symphony Orch...	0.99
139	Cláudio Zoli	0.99
140	Chris Cornell	0.99
141	David Coverdale	0.99
142	Def Leppard	0.99
143	Dennis Chambers	0.99



Βλέπουμε ότι η εφαρμογή επέστρεψε όλους τους καλλιτέχνες και συνεπώς έχουμε 144 αποτελέσματα.

Η λογική σχεδίαση των υπόλοιπων Stored procedures καθώς και ο τρόπος εκτέλεσης είναι ο ίδιος.

2) Βρες τα τοπ 10 τραγούδια στις προτιμήσεις των πελατών για συγκεκριμένο χρονικό διάστημα.

Παρακάτω παρουσιάζεται το Stored Procedure “spTop10Tracks” που περιλαμβάνει τον κώδικα SQL για την υλοποίηση του ερωτήματος. Το sum(il.TrackId) που παρουσιάζεται παρακάτω αθροίζει το πλήθος των τεμαχίων ανά τραγούδι, συνεπώς πόσες φορές αγοράστηκε το συγκεκριμένο τραγούδι.

spTop10Tracks.sql -...-JVJ5FIF\Panos (52))

```
ALTER PROC spTop10Tracks
@DateFrom date = NULL,
@DateTo date = NULL
AS
BEGIN
BEGIN TRANSACTION
    SELECT TOP(10) t.name 'Track Name', sum(il.TrackId) 'Total Purchases'
    FROM track t, InvoiceLine il, Invoice i
    WHERE il.TrackId = t.Trackid
        AND il.invoiceid = i.invoiceid
        AND(((NullIf(@DateFrom, '') IS NULL)  OR CAST(i.InvoiceDate AS DATE) >= @DateFrom)
        AND ((NullIf(@DateTo, '') IS NULL)  OR CAST(i.InvoiceDate AS DATE) <= @DateTo))
    GROUP BY t.name
    ORDER BY 'Total Purchases' desc
COMMIT TRANSACTION
END
GO 1
```

Εκτελούμε την παραπάνω διαδικασία και μας ενδιαφέρουν τα τοπ 10 τραγούδια μεταξύ των ετών 2010 και 2013 και έχουμε τα ανάλογα αποτελέσματα.

Parameter	Data Type	Output Parameter	Pass Null Value	Value
@DateFrom	date	No	☐	'2010'
@DateTo	date	No	☐	'2013'

Results | Messages

	Track Name	Total Purchases
1	String Quartet No. 12 in C Minor, D. 703 "Quartetts...	7000
2	Music for the Funeral of Queen Mary: VI. "Thou Kn...	6976
3	Suite No. 3 in D, BWV 1068: III. Gavotte I & II	6964
4	Rehab	6910
5	Symphonie Fantastique, Op. 14: V. Songe d'une nui...	6892
6	Scheherazade, Op. 35: I. The Sea and Sindbad's S...	6864
7	Branch Closing	6634
8	War Pigs	6560
9	Beautiful Boy	6536
10	Gimme Some Truth	6532

3) Βρες ποια είδη μουσικής είναι διαχρονικά πρώτα στις προτιμήσεις των πελατών.

Παρακάτω παρουσιάζεται το Stored Procedure “spTopGenres” που περιλαμβάνει τον κώδικα SQL για την υλοποίηση του ερωτήματος.

spTopGenres.sql - D...JVJ5FIF\Panos (53))

```
ALTER PROC spTopGenres
AS
BEGIN
BEGIN TRANSACTION
    SELECT g.name 'Genre', sum(il.TrackId) 'Total Purchases'
    FROM genre g, track t, InvoiceLine il, Invoice i
    WHERE g.GenreId = t.GenreId
    and il.TrackId = t.Trackid
    and il.invoiceid = i.invoiceid
    GROUP BY g.name
    ORDER BY 'Total Purchases' desc
COMMIT TRANSACTION
END
GO 1
```

Εκτελούμε την παραπάνω διαδικασία και έχουμε τα ανάλογα αποτελέσματα.

Results | Messages

	Genre	Total Purchases
1	Rock	1476471
2	Latin	478692
3	Alternative & Punk	437512
4	Metal	392155
5	TV Shows	141886
6	Classical	141791
7	Blues	89901
8	Drama	87737
9	R&B/Soul	86591
10	Jazz	75369
11	Pop	75016
12	Sci Fi & Fantasy	63947
13	Alternative	47469
14	Hip Hop/Rap	46690
15	Reggae	42454
16	Electronica/Dance	30584
17	Comedy	29135
18	Soundtrack	26774
19	World	24369
20	Science Fiction	16978
21	Heavy Metal	15267
22	Easy Listening	10443
23	Bossa Nova	9797
24	Rock And Roll	697

4) Δημιούργησε μια αναφορά όπου θα εμφανίζονται τα στοιχεία επικοινωνίας των πελατών που έχουν κάνει παραγγελίες και ο αντίστοιχος συνολικός τους τζίρος, ταξινομημένα με φθίνουσα σειρά ανά τζίρο. Ο χρήστης θα πρέπει να ορίζει τα εξής κριτήρια για την δημιουργία της αναφοράς: Χρονικό διάστημα παραγγελιών Ημερομηνία Από – Έως.

Παρακάτω παρουσιάζεται το Stored Procedure “spCustomerContactDetails” που περιλαμβάνει τον κώδικα SQL για την υλοποίηση του ερωτήματος.

spCustomerContact...VJ5FIF\Panos (64))

```
ALTER PROC spCustomerContactDetails
@DateFrom date = NULL,
@DateTo date = NULL
AS
BEGIN
BEGIN TRANSACTION
    SELECT c.CustomerId, c.Phone, c.Fax, c.Email, i.Total 'Total Income'
    FROM Customer c, Invoice i
    WHERE c.CustomerId = i.CustomerId
        AND(((NullIf(@DateFrom, '') IS NULL)  OR CAST(i.InvoiceDate AS DATE) >= @DateFrom)
        AND ((NullIf(@DateTo, '') IS NULL)  OR CAST(i.InvoiceDate AS DATE) <= @DateTo))
    ORDER BY i.Total desc
COMMIT TRANSACTION
END
GO 1
```

Εκτελούμε την παραπάνω διαδικασία και μας ενδιαφέρουν στοιχεία επικοινωνίας των πελλατών με χρονικό διάστημα παραγγελιών μέχρι και το 2011

Parameter	Data Type	Output Parameter	Pass Null Value	Value
@DateFrom	date	No	☐	
@DateTo	date	No	☐	'2011'

Results | Messages

	CustomerId	Phone	Fax	Email	Total Income
1	45	NULL	NULL	ladislav_kovacs@apple.hu	21.86
2	7	+43 01 5134505	NULL	astrid.gruber@apple.at	18.86
3	57	+56 (0)2 635 4444	NULL	luisrojas@yahoo.cl	17.91
4	24	+1 (312) 332-3232	NULL	fralston@gmail.com	15.86
5	3	+1 (514) 721-4711	NULL	ftremblay@gmail.com	13.86
6	41	+33 04 78 30 30 30	NULL	marc.dubois@hotmail.com	13.86
7	20	+1 (650) 644-3358	NULL	dmiller@comcast.com	13.86
8	28	+1 (801) 531-7272	NULL	jubarnett@gmail.com	13.86
9	11	+55 (11) 3055-3278	+55 (11) 3055-8131	alero@uol.com.br	13.86
10	49	+48 22 828 37 39	NULL	stanisław.wójcik@wp.pl	13.86
11	23	+1 (617) 522-1333	NULL	johngordon22@yahoo.com	13.86
12	2	+49 0711 2842222	NULL	leonekohler@surfeu.de	13.86
13	40	+33 01 47 42 71 71	NULL	dominiquelefebvre@gmail.com	13.86
14	19	+1 (408) 996-1010	+1 (408) 996-1011	tgoyer@apple.com	13.86
15	57	+56 (0)2 635 4444	NULL	luisrojas@yahoo.cl	13.86
16	36	+49 030 26550280	NULL	hannah.schneider@yahoo.de	13.86
17	15	+1 (604) 688-2255	+1 (604) 688-8756	jenniferp@rogers.ca	13.86
18	53	+44 020 7976 5722	NULL	phil.hughes@gmail.com	13.86
19	32	+1 (204) 452-6452	NULL	aaronmitchell@yahoo.ca	13.86
20	58	+91 0124 39883988	NULL	manoj.pareek@rediff.com	13.86
21	37	+49 069 40598889	NULL	fzimmermann@yahoo.de	13.86
22	16	+1 (650) 253-0000	+1 (650) 253-0000	fharris@google.com	13.86
23	54	+44 0131 315 3300	NULL	steve.murray@yahoo.uk	13.86
24	33	+1 (867) 920-2233	NULL	ellie.sullivan@shaw.ca	13.86

5) Δημιούργησε μια αναφορά που θα δείχνει το Id της παραγγελίας (InvoiceID), το ονοματεπώνυμο του πελάτη που έκανε τη παραγγελία και το ονοματεπώνυμο του υπαλλήλου που σχετίζεται με τον αντίστοιχο πελάτη. Εμφάνισε εκείνες τις παραγγελίες που έγιναν εντός ενός ημερολογιακού διαστήματος. Ο χρήστης θα μπορεί να ορίζει τα εξής κριτήρια για την δημιουργία της αναφοράς: Χρονικό διάστημα παραγγελιών Ημερομηνία Από - Έως, Επιλογή πελατών από μια λίστα πελάτων του ηλεκτρονικού δισκοπωλείου, Υπάλληλος που σχετίζεται με τη παραγγελία.

Παρακάτω παρουσιάζεται το Stored Procedure “spOrderDetails” που περιλαμβάνει τον κώδικα SQL για την υλοποίηση του ερωτήματος.

spOrderDetails.sql...-JVJ5FIF\Panos (53))

```
ALTER PROC spOrderDetails
@DateFrom date = NULL,
@DateTo date = NULL,
@EmployeeFN nvarchar(50) = NULL,
@EmployeeLN nvarchar(50) = NULL,
@CustomerFN nvarchar(50) = NULL,
@CustomerLN nvarchar(50) = NULL
AS
BEGIN
BEGIN TRANSACTION
    SELECT i.InvoiceId, c.FirstName 'Customer FirstName', c.LastName 'Customer LastName',
        e.FirstName 'Employee FirstName', e.LastName 'Employee LastName'
    FROM Customer c, Invoice i, Employee e
    WHERE c.CustomerId = i.CustomerId
        AND c.CustomerId = e.ReportsTo
        AND (((NullIf(@DateFrom, '') IS NULL)  OR CAST(i.InvoiceDate AS DATE) >= @DateFrom)
        AND ((NullIf(@DateTo, '') IS NULL)  OR CAST(i.InvoiceDate AS DATE) <= @DateTo))
        AND (((NullIf(@EmployeeFN, '') IS NULL)  OR CAST(e.FirstName AS NVARCHAR(50)) = @EmployeeFN)
        AND ((NullIf(@EmployeeLN, '') IS NULL)  OR CAST(e.LastName AS NVARCHAR(50)) = @EmployeeLN))
        AND (((NullIf(@CustomerFN, '') IS NULL)  OR CAST(c.FirstName AS NVARCHAR(50)) = @CustomerFN)
        AND ((NullIf(@CustomerLN, '') IS NULL)  OR CAST(c.LastName AS NVARCHAR(50)) = @CustomerLN))
COMMIT TRANSACTION
END
GO 1
```

Εκτελούμε την παραπάνω διαδικασία και μας ενδιαφέρουν οι παραγγελίες εκείνες που έχουν μικρό όνομα πελάτη ‘Hellena’ και επώνυμο υπαλλήλου ‘King’.

Parameter	Data Type	Output Parameter	Pass Null Value	Value
@DateFrom	date	No	☐	
@DateTo	date	No	☐	
@EmployeeFN	nvarchar(50)	No	☐	
@EmployeeLN	nvarchar(50)	No	☐	King
@CustomerFN	nvarchar(50)	No	☐	Helena
@CustomerLN	nvarchar(50)	No	☐	

Results Messages

	InvoiceId	Customer FirstName	Customer LastName	Employee FirstName	Employee LastName
1	46	Helena	Holý	Robert	King
2	175	Helena	Holý	Robert	King
3	198	Helena	Holý	Robert	King
4	220	Helena	Holý	Robert	King
5	272	Helena	Holý	Robert	King
6	393	Helena	Holý	Robert	King
7	404	Helena	Holý	Robert	King

Β) Δημιουργία Stored Procedure InvoiceStatistics

Παρακάτω παρουσιάζεται το Stored Procedure “spInvoiceStatistics” που περιλαμβάνει τον κώδικα SQL για την υλοποίηση του του Β’ σταδίου.

InvoiceStatistics.sq...-JVJ5FIF\Panos (67))

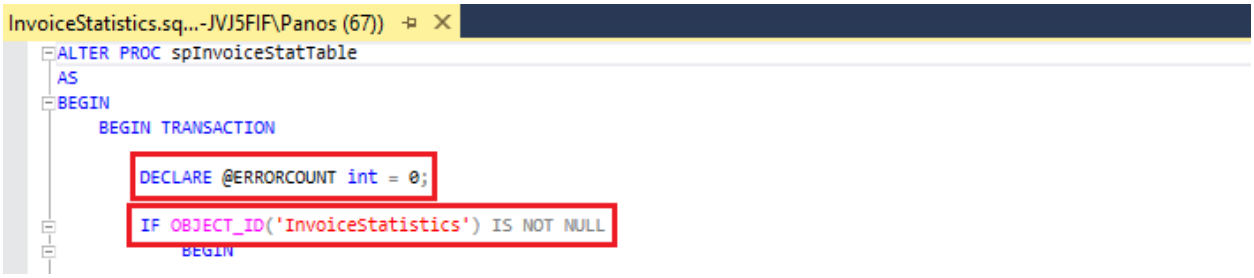
```
ALTER PROC spInvoiceStatTable
AS
BEGIN
    BEGIN TRANSACTION

        DECLARE @ERRORCOUNT int = 0;

        IF OBJECT_ID('InvoiceStatistics') IS NOT NULL
            BEGIN

                PRINT ''
                PRINT 'TABLE EXIST'

                --!Error Check for DELETE ROWS
                BEGIN TRY
                    DELETE FROM InvoiceStatistics;
                    PRINT ''
                    PRINT 'ALL ROWS DELETED'
                END TRY
                BEGIN CATCH
                    SET @ERRORCOUNT = -2;
                END CATCH

                --!Error Check for INSERT ROWS
                BEGIN TRY
                    INSERT INTO InvoiceStatistics (GenreId, TrackId, TotalTrackCharge, TimeCreated)
                    SELECT t.GenreId, il.TrackId, il.UnitPrice*il.Quantity, i.InvoiceDate
                    FROM InvoiceLine il, Track t, Invoice i
                    WHERE il.TrackId = t.Trackid
                    and il.invoiceid = i.invoiceid;
                    PRINT ''
                    PRINT 'ALL ROWS INSERTED'
                END TRY
                BEGIN CATCH
                    SET @ERRORCOUNT = -3;
                END CATCH

            END
            ELSE
                BEGIN

                PRINT ''
                PRINT 'TABLE DOES NOT EXIST'

                --!Error Check for CREATE TABLE
                BEGIN TRY
                    CREATE TABLE InvoiceStatistics (
                                GenreId int,
                                TrackId int,
                                TotalTrackCharge numeric(10,2),
                                TimeCreated datetime);
                    PRINT ''
                    PRINT 'TABLE CREATED'
                END TRY
                BEGIN CATCH
                    SET @ERRORCOUNT = -1;
                END CATCH

                --!Error Check for INSERT ROWS
                BEGIN TRY
```

```
            --!Error Check for INSERT ROWS
            BEGIN TRY
                INSERT INTO InvoiceStatistics (GenreId, TrackId, TotalTrackCharge, TimeCreated)
                SELECT t.GenreId, il.TrackId, il.UnitPrice*il.Quantity, i.InvoiceDate
                FROM InvoiceLine il, Track t, Invoice i
                WHERE il.TrackId = t.Trackid
                and il.invoiceid = i.invoiceid;
                PRINT ''
                PRINT 'ALL ROWS INSERTED'
            END TRY
            BEGIN CATCH
                SET @ERRORCOUNT = -3;
            END CATCH

            END

    IF(@ERRORCOUNT = 0)
        BEGIN
            SELECT * FROM InvoiceStatistics
            COMMIT TRANSACTION
        END
    ELSE
        BEGIN
            IF(@ERRORCOUNT = -1) RAISERROR('-1',16,1)
            IF(@ERRORCOUNT = -2) RAISERROR('-2',16,1)
            IF(@ERRORCOUNT = -3) RAISERROR('-3',16,1)
            ROLLBACK TRANSACTION
        END
END
GO 1
```

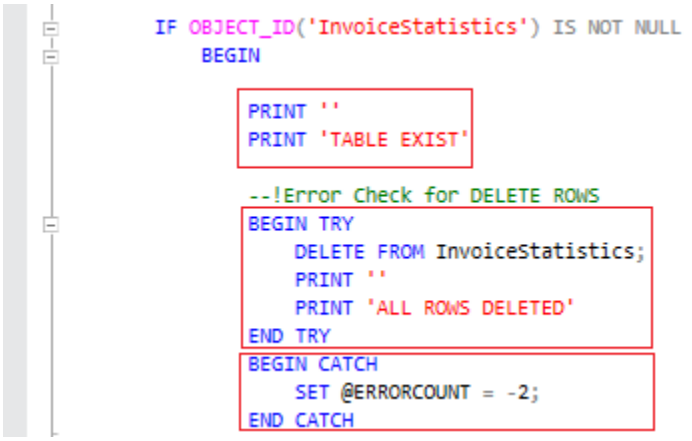
Όλη η διαδικασία περιέχεται μέσα σε ένα Transaction και αυτό έχει ως αποτέλεσμα την υλοποίηση όλων των πτυχών της διαδικασίας (Commit) ή καμία απολύτως εκτέλεση και επαναφορά στο σημείο πριν ακριβώς τη εκτέλεσή της (Rollback).

```
AS
BEGIN
    BEGIN TRANSACTION

        DECLARE @ERRORCOUNT int = 0;


    IF(@ERRORCOUNT = 0)
        BEGIN
            SELECT * FROM InvoiceStatistics
            COMMIT TRANSACTION
        END
    ELSE
        BEGIN
            IF(@ERRORCOUNT = -1) RAISERROR('-1',16,1)
            IF(@ERRORCOUNT = -2) RAISERROR('-2',16,1)
            IF(@ERRORCOUNT = -3) RAISERROR('-3',16,1)
            ROLLBACK TRANSACTION
        END
END
GO 1
```

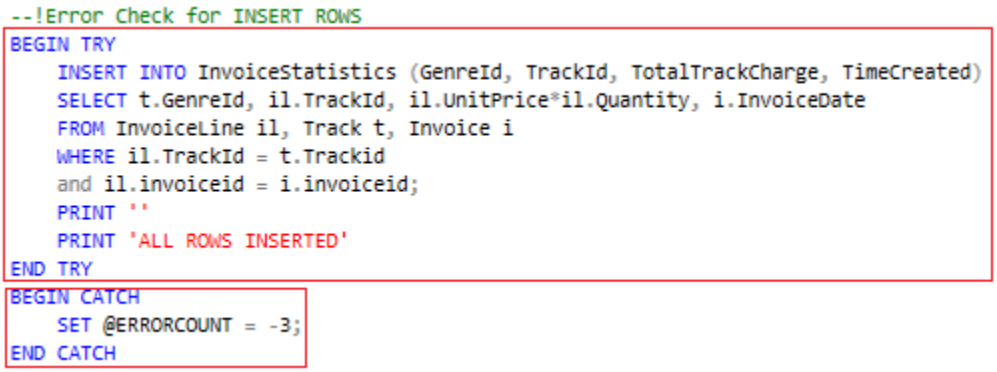
Η διαδικασία ξεκινάει με την αρχικοποίηση της μεταβλητής @ErrorCount, σκοπός της οποίας είναι να αποθηκεύει τον αναγνωριστικό αριθμό Error όπως ακριβώς απαιτεί η εκφώνηση (π.χ. αποτυχία δημιουργία πίνακα @ErrorCount = -1). Έπειτα ξεκινάει η εντολή IF μέσω της οποίας ελέγχουμε αν ο πίνακας δεν είναι κενός και κατεπέκταση αν υπάρχει.



Εάν ο συγκεκριμένος έλεγχος είναι αληθής, ο χρήστης ενημερώνεται με κατάλληλο μήνυμα ότι ο πίνακας υπάρχει και συνεπώς έχει περιεχόμενο, άρα η διαδικασία προχωράει στη διαγραφή όλων των γραμμών του και στην ενημέρωση του χρήστη με κατάλληλο μήνυμα. Σε περίπτωση σφάλματος κατά τη διαγραφή των γραμμών η διαδικασία εντός της Try αναιρείτε και στη θέση της εκτελείτε το μέρος της Catch με αποτέλεσμα το @ErroCount να γίνει ίσο με -2 όπως απαιτεί η εκφώνηση.



Έπειτα ακολουθεί το ίδιο μοτίβο καθώς προχωράμε στην εισαγωγή των νέων γραμμών από τους πίνακες Invoice, InvoiceLine καθώς όμως και του πίνακα Track καθώς είναι απαραίτητος για να έχουμε πρόσβαση στο χαρακτηριστικό GenreId. Σε περίπτωση σφάλματος το @ErrorCount γίνεται ίσο με -3.



Στην περίπτωση όμως που η αρχική IF βγει ψευδής, ο πίνακας δεν υπάρχει στο σύστημα και ο χρήστης ενημερώνεται με κατάλληλο μήνυμα. Συνεπώς η διαδικασία προχωρά στη δημιουργία του πίνακα InvoiceStatistics ενημερώνοντας το χρήστη και όπως πριν στην περίπτωση σφάλματος, το συγκεκριμένο Block δεν εκτελείτε και στη θέση του εκτελείτε το Catch block που θέτει στο @ErrorCount τη τιμή -1.

```
ELSE
    BEGIN

    PRINT ''
    PRINT 'TABLE DOES NOT EXIST'

    --!Error Check for CREATE TABLE
    BEGIN TRY
        CREATE TABLE InvoiceStatistics (
                    GenreId int,
                    TrackId int,
                    TotalTrackCharge numeric(10,2),
                    TimeCreated datetime);
        PRINT ''
        PRINT 'TABLE CREATED'
    END TRY
    BEGIN CATCH
        SET @ERRORCOUNT = -1;
    END CATCH
```

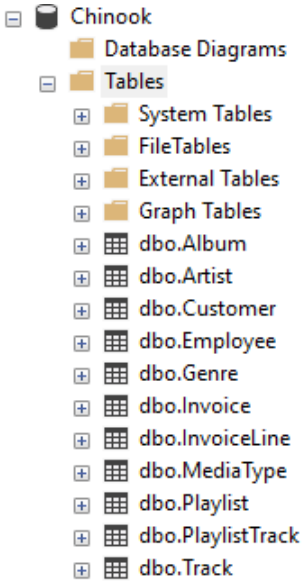
Έπειτα ακολούθως της δημιουργίας του πίνακα, το επόμενο block προχωρά στην εισαγωγή νέων γραμμών. Δεν χρειάζεται να πούμε αρκετά πράγματα σε αυτό το σημείο καθώς το συγκεκριμένο block είναι πανομοιότυπο με εκείνο της εισαγωγής στη περίπτωση που η IF είναι αληθής, δηλαδή που ο πίνακας υπήρχε εξαρχής.

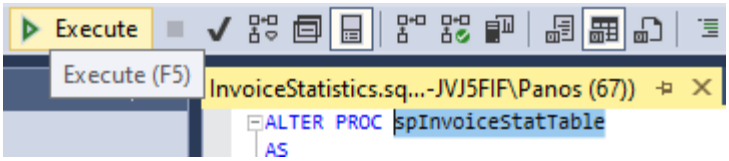
```
    --!Error Check for INSERT ROWS
    BEGIN TRY
        INSERT INTO InvoiceStatistics (GenreId, TrackId, TotalTrackCharge, TimeCreated)
        SELECT t.GenreId, il.TrackId, il.UnitPrice*il.Quantity, i.InvoiceDate
        FROM InvoiceLine il, Track t, Invoice i
        WHERE il.TrackId = t.Trackid
        and il.invoiceid = i.invoiceid;
        PRINT ''
        PRINT 'ALL ROWS INSERTED'
    END TRY
    BEGIN CATCH
        SET @ERRORCOUNT = -3;
    END CATCH

    END
```

Τέλος ανεξαρτήτως ποιου block της IF εκτελέστηκε στο τέλος γίνεται ο έλεγχος για αν πρέπει να Commit ή Rollback για το συγκεκριμένο Transaction. Σε περίπτωση που η συνθήκη @ErrorCount = 0 είναι αληθής, τότε προχωράμε στη εμφάνιση του περιεχομένου του πίνακα InvoiceStatistics και εκτελείτε Commit για το συγκεκριμένο Transaction καθώς όλα τα Block εκτελέστηκαν με επιτυχία χωρίς τη παρουσία κάποιου Error. Σε αντίθετη περίπτωση όμως που το @ErrorCount έχει λάβει οποιαδήποτε άλλη τιμή τότε ο χρήστης ενημερώνεται με Custom Error μήνυμα ανάλογα με τη φύση του, δηλαδή τη τιμή του ErrorCount. Έπειτα εκτελείτε Rollback και όλες οι διαδικασίες που έλαβαν χώρα εντός του Transaction αναιρούνται.

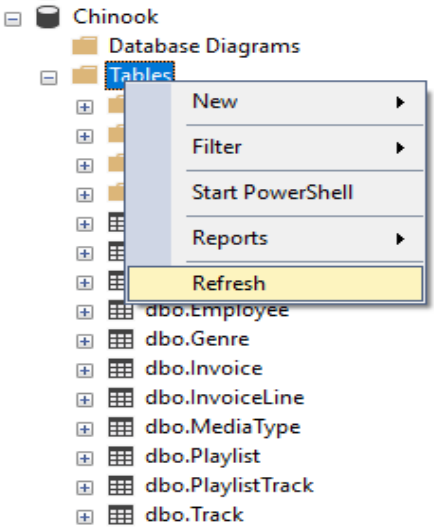
```
IF(@ERRORCOUNT = 0)
    BEGIN
        SELECT * FROM InvoiceStatistics
        COMMIT TRANSACTION
    END
ELSE
    BEGIN
        IF(@ERRORCOUNT = -1) RAISERROR('-1',16,1)
        IF(@ERRORCOUNT = -2) RAISERROR('-2',16,1)
        IF(@ERRORCOUNT = -3) RAISERROR('-3',16,1)
        ROLLBACK TRANSACTION
    END
END
GO 1
```

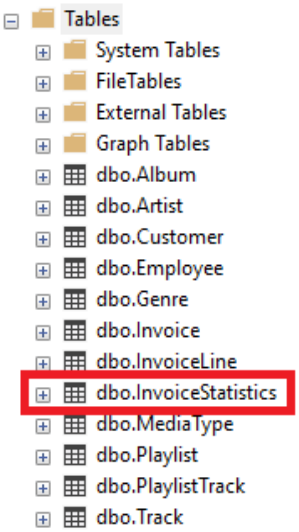
Προχωράμε λοιπόν στην εκτέλεση της συγκεκριμένης διαδικασίας. Σε αυτό το στάδιο η βάσει δεδομένων είναι ακριβώς στη μορφή που τη λάβαμε και όπως βλέπουμε ο πίνακας InvoiceStatistics δεν υπάρχει. Κάνουμε highlight λοιπόν το όνομα του Stored Procedure για συντομία και πατάμε εκτέλεση.





Κάνοντας Refresh βλέπουμε ότι ο πίνακας InvoiceStatistics δημιουργήθηκε.





Και όπως βλέπουμε έχουμε τα ανάλογα αποτελέσματα. Εφόσον ο πίνακας δημιουργείτε και γεμίζει με στοιχεία από τους πίνακες Invoice, InvoiceLine και Track. Επίσης ο χρήστης ενημερώνεται για τα αποτελέσματα του Transaction στο όπως φαίνεται στο πίνακα Messages.

Results | Messages

	GenreId	TrackId	TotalTrackCharge	TimeCreated
1	1	1	0.99	2010-04-13 00:00:00.000
2	1	2	0.99	2009-01-01 00:00:00.000
3	1	2	0.99	2011-07-25 00:00:00.000
4	1	3	0.99	2012-11-01 00:00:00.000
5	1	4	0.99	2009-01-01 00:00:00.000
6	1	5	0.99	2010-04-13 00:00:00.000
7	1	6	0.99	2009-01-02 00:00:00.000
8	1	8	0.99	2009-01-02 00:00:00.000
9	1	8	0.99	2011-07-25 00:00:00.000
10	1	9	0.99	2010-04-13 00:00:00.000
11	1	9	0.99	2012-11-01 00:00:00.000
12	1	10	0.99	2009-01-02 00:00:00.000
13	1	12	0.99	2009-01-02 00:00:00.000
14	1	13	0.99	2010-04-13 00:00:00.000
15	1	14	0.99	2011-07-25 00:00:00.000

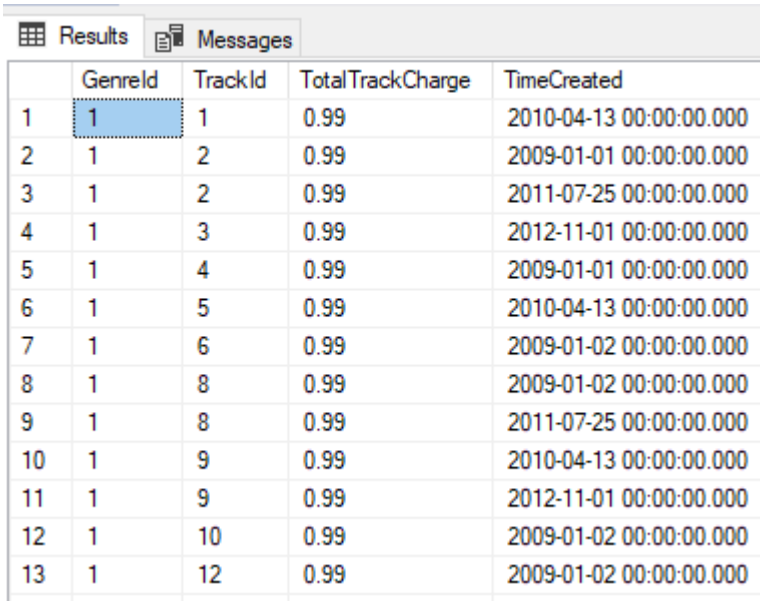
Results | Messages

```
TABLE DOES NOT EXIST

TABLE CREATED

(2240 rows affected)

ALL ROWS INSERTED

(2240 rows affected)

Completion time: 2020-08-07T13:44:15.6162761+03:00
```

2...	24	3481	0.99	2011-07-22 00:00:00.000
2...	24	3482	0.99	2010-04-11 00:00:00.000
2...	24	3482	0.99	2012-11-01 00:00:00.000
2...	24	3484	0.99	2010-04-11 00:00:00.000
2...	24	3485	0.99	2011-07-22 00:00:00.000
2...	24	3486	0.99	2010-04-12 00:00:00.000
2...	24	3488	0.99	2010-04-12 00:00:00.000
2...	24	3488	0.99	2012-11-01 00:00:00.000
2...	24	3489	0.99	2011-07-22 00:00:00.000
2...	24	3490	0.99	2010-04-12 00:00:00.000
2...	24	3492	0.99	2010-04-12 00:00:00.000
2...	24	3493	0.99	2011-07-22 00:00:00.000
2...	24	3494	0.99	2012-11-01 00:00:00.000
2...	24	3496	0.99	2010-04-13 00:00:00.000
2...	24	3499	0.99	2011-07-25 00:00:00.000
2...	24	3500	0.99	2010-04-13 00:00:00.000
2...	24	3500	0.99	2012-11-01 00:00:00.000

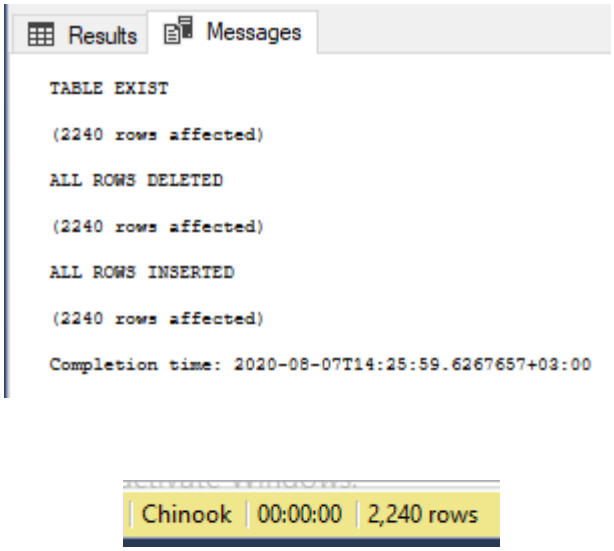
Chinook | 00:00:00 | 2,240 rows

Επαναλαμβάνουμε πάλι την ίδια διαδικασία δηλαδή την εκτέλεση της διαδικασίας και όπως βλέπουμε έχομε τα ίδια αποτελέσματα στα Results καθώς απλά όλες οι εγγραφές διεγράφησαν και όπως βλέπουμε στο Messages ο χρήστης ενημερώνεται καταλλήλως.



Results | Messages

	GenreId	TrackId	TotalTrackCharge	TimeCreated
1	1	1	0.99	2010-04-13 00:00:00.000
2	1	2	0.99	2009-01-01 00:00:00.000
3	1	2	0.99	2011-07-25 00:00:00.000
4	1	3	0.99	2012-11-01 00:00:00.000
5	1	4	0.99	2009-01-01 00:00:00.000
6	1	5	0.99	2010-04-13 00:00:00.000
7	1	6	0.99	2009-01-02 00:00:00.000
8	1	8	0.99	2009-01-02 00:00:00.000
9	1	8	0.99	2011-07-25 00:00:00.000
10	1	9	0.99	2010-04-13 00:00:00.000
11	1	9	0.99	2012-11-01 00:00:00.000
12	1	10	0.99	2009-01-02 00:00:00.000
13	1	12	0.99	2009-01-02 00:00:00.000



Results | Messages

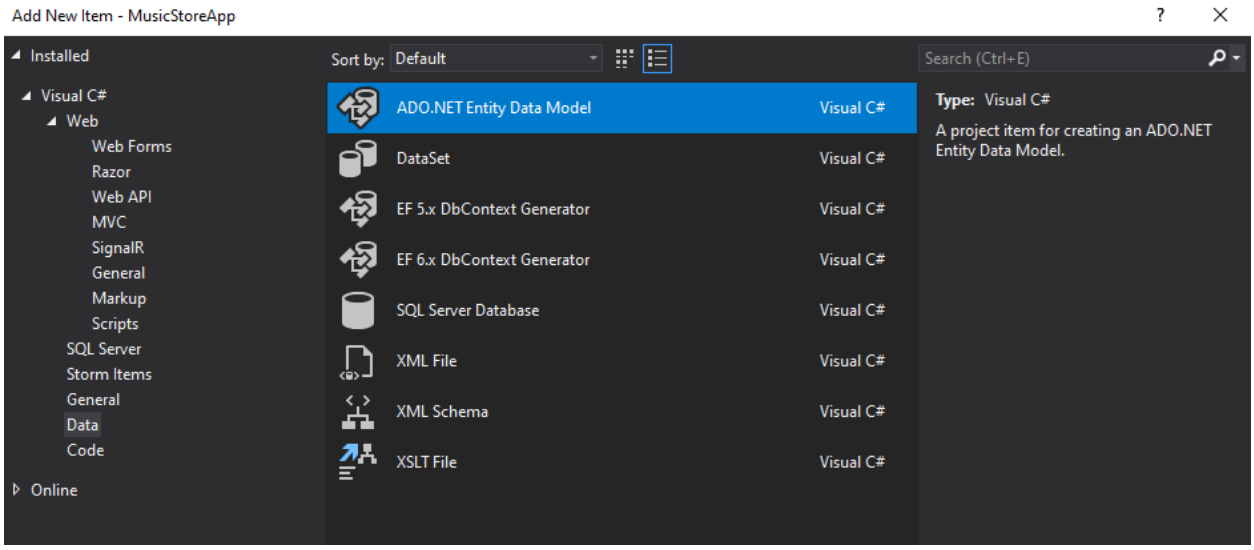
```
TABLE EXIST

(2240 rows affected)

ALL ROWS DELETED

(2240 rows affected)

ALL ROWS INSERTED

(2240 rows affected)

Completion time: 2020-08-07T14:25:59.6267657+03:00
```

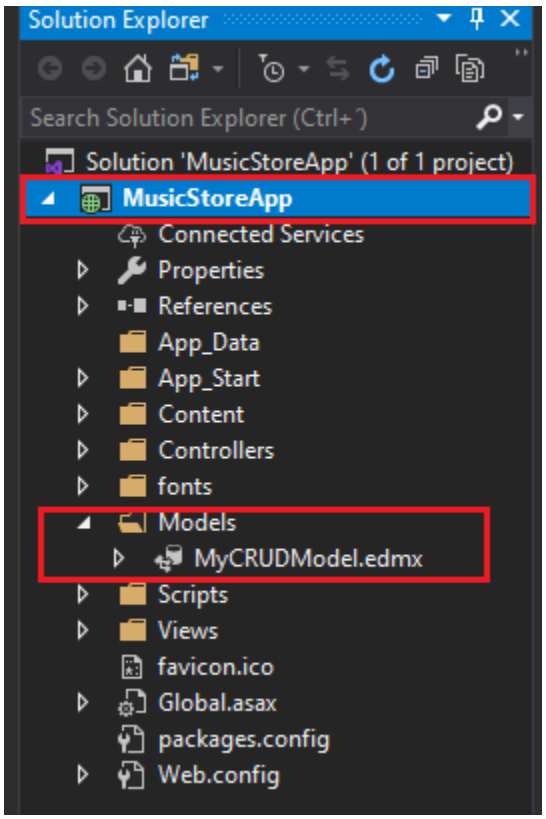

Chinook | 00:00:00 | 2,240 rows

Όλα τα Stored Procedures που παρουσιάστηκαν βρίσκονται στο φάκελο “Stored Procedures” εντός του παραδοτέου στους φακέλους “Στάδιο Α” και “Στάδιο Β” αντίστοιχα.

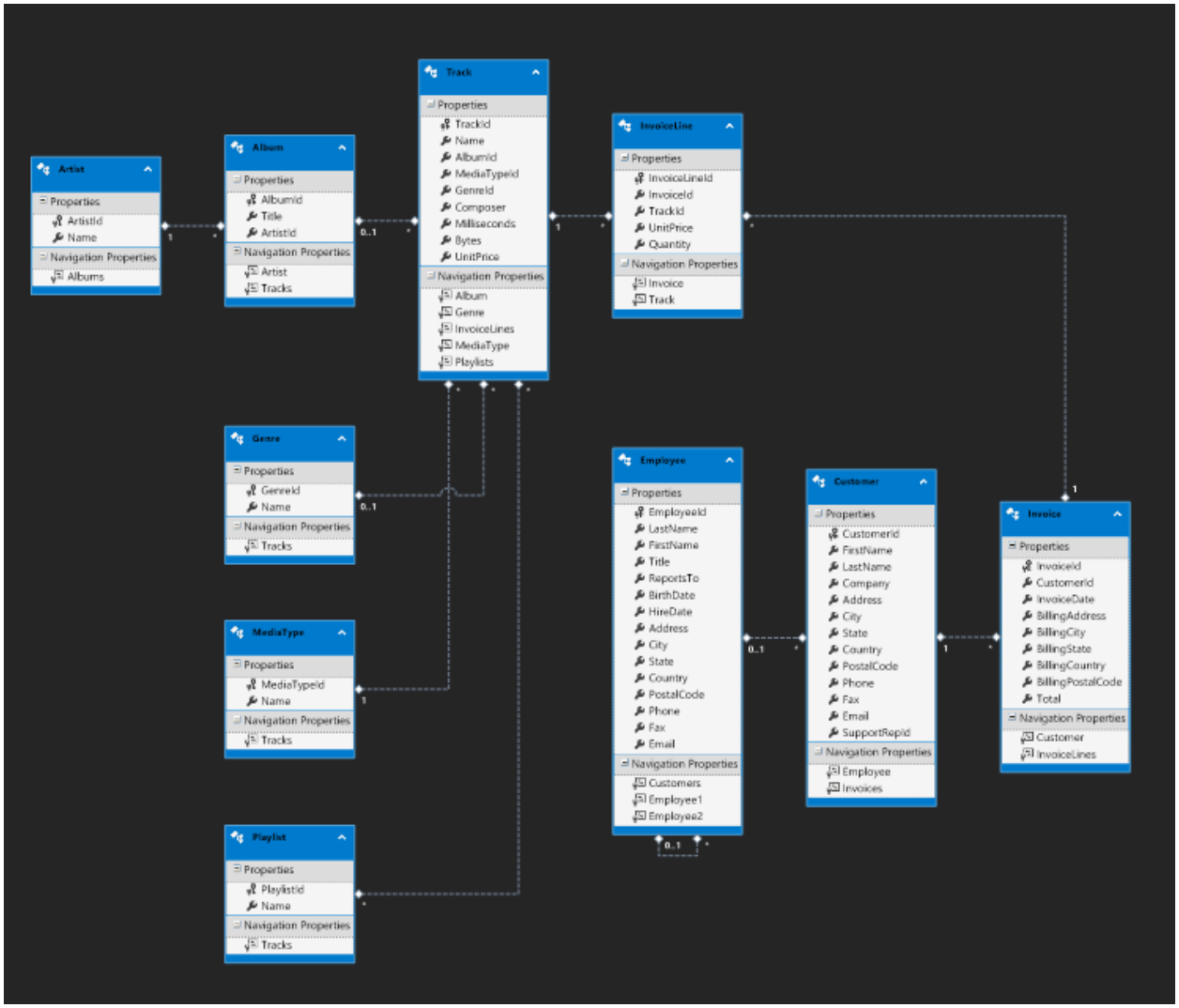
C) Ανάπτυξη Διαδικτυακής Εφαρμογής

Ακολουθώντας τα βήματα από τον οδηγό της Microsoft που φαίνεται στο ακόλουθο Link https://docs.microsoft.com/en-us/aspnet/mvc/overview/getting-started/database-first-development/ προχωρήσαμε στη δημιουργία του web application “MusicStoreApp” και στην δημιουργία της κλάσης MyCrudModel” που αναπαριστά τις οντότητες της βάσεις δεδομένων που διαχειριζόμαστε.

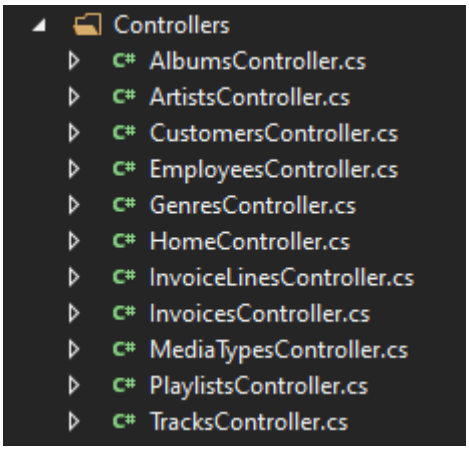




Το μοντέλο οντοτήτων παρουσιάζεται παρακάτω.



Στη συνέχεια προχωρήσαμε στη δημιουργία των Controllers για τις εκάστοτε οντότητες, τις μεθόδους δηλαδή εισαγωγής, διαγραφής, επεξεργασίας για όλους τους πίνακες της βάσης όπως φαίνεται παρακάτω.



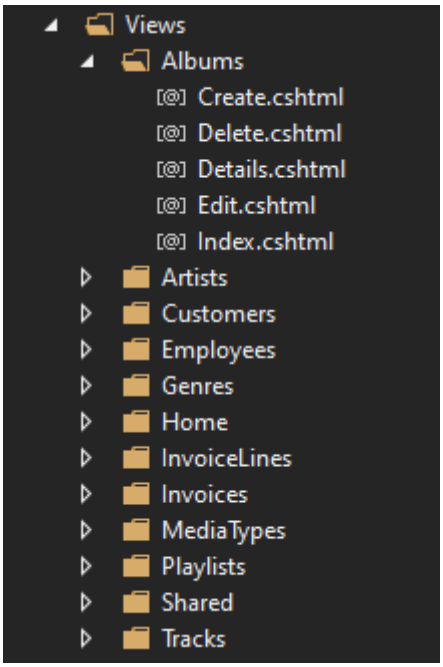
AlbumsController.cs

MusicStoreApp

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Linq;
using System.Threading.Tasks;
using System.Net;
using System.Web;
using System.Web.Mvc;
using MusicStoreApp.Models;

namespace MusicStoreApp.Controllers
{
    0 references
    public class AlbumsController : Controller
    {
        private ChinookEntities db = new ChinookEntities();

        // GET: Albums
        0 references
        public async Task<ActionResult> Index()
        {
            var albums = db.Albums.Include(a => a.Artist);
            return View(await albums.ToListAsync());
        }

        // GET: Albums/Details/5
        0 references
        public async Task<ActionResult> Details(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Album album = await db.Albums.FindAsync(id);
            if (album == null)
            {
                return HttpNotFound();
            }
            return View(album);
        }

        // GET: Albums/Create
```

(Δυστυχώς δεν καταφέραμε να υλοποιήσουμε τη σχεδίαση master detail για τους πίνακες Invoice – InvoiceLine οπότε για λόγους τυπικότητας απλά δημιουργήσαμε αυτοτελείς controllers όπως και στους άλλους πίνακες. Επίσης αντιμετωπίσαμε πρόβλημα και δεν καταφέραμε να συνδέσουμε τα Stored Procedures με τις οντότητες του μοντέλου της εφαρμογής, καθώς τα αποτελέσματα των procedures προκύπτουν από διαφορετικούς πίνακες και όχι μόνο από έναν πράγμα που μας δυσκόλεψε καθώς κάθε procedure αντιστοιχίζετε σε μια οντότητα.)

Έπειτα Δημιουργήσαμε τα Views για την κάθε οντότητα, το γραφικό περιβάλλον δηλαδή που θα εμφανίζετε στον χρήστη και μέσω του οποίου θα μπορεί να αλληλοεπιδρά με την βάση δεδομένων.




```
@model MusicStoreApp.Models.Album

@{
    ViewBag.Title = "Create";
}

<h2>Create</h2>


@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
        <h4>Album</h4>
        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        <div class="form-group">
            @Html.LabelFor(model => model.Title, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Title, new { htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Title, "", new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.ArtistId, "ArtistId", htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.DropDownList("ArtistId", null, htmlAttributes: new { @class = "form-control" })
                @Html.ValidationMessageFor(model => model.ArtistId, "", new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <input type="submit" value="Create" class="btn btn-default" />
            </div>
        </div>
    </div>
}

<div>
    @Html.ActionLink("Back to List", "Index")
</div>
```

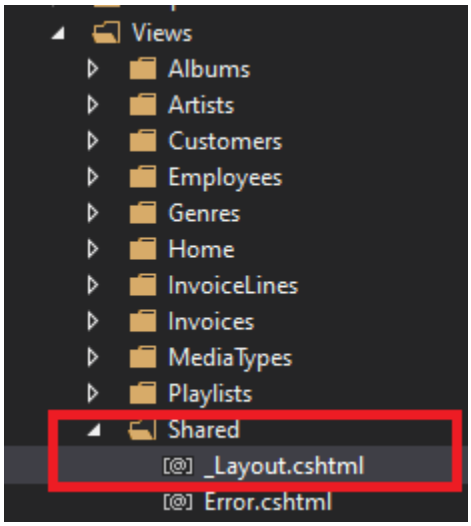
```
@model MusicStoreApp.Models.Album

@{
    ViewBag.Title = "Delete";
}

<h2>Delete</h2>

<h3>Are you sure you want to delete this?</h3>
<div>
    <h4>Album</h4>
    <hr />
    <dl class="dl-horizontal">
        <dt>
            @Html.DisplayNameFor(model => model.Title)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Title)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Artist.Name)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Artist.Name)
        </dd>

    </dl>

    @using (Html.BeginForm()) {
        @Html.AntiForgeryToken()

        <div class="form-actions no-color">
            <input type="submit" value="Delete" class="btn btn-default" /> |
            @Html.ActionLink("Back to List", "Index")
        </div>
    }
</div>
```

```
@model MusicStoreApp.Models.Album

@{
    ViewBag.Title = "Details";
}

<h2>Details</h2>

<div>
    <h4>Album</h4>
    <hr />
    <dl class="dl-horizontal">
        <dt>
            @Html.DisplayNameFor(model => model.Title)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Title)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Artist.Name)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Artist.Name)
        </dd>

    </dl>
</div>
<p>
    @Html.ActionLink("Edit", "Edit", new { id = Model.AlbumId }) |
    @Html.ActionLink("Back to List", "Index")
</p>
```

```cshtml
@model MusicStoreApp.Models.Album

@{
    ViewBag.Title = "Edit";
}

<h2>Edit</h2>


@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
        <h4>Album</h4>
        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        @Html.HiddenFor(model => model.AlbumId)

        <div class="form-group">
            @Html.LabelFor(model => model.Title, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Title, new { htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Title, "", new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.ArtistId, "ArtistId", htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.DropDownList("ArtistId", null, htmlAttributes: new { @class = "form-control" })
                @Html.ValidationMessageFor(model => model.ArtistId, "", new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <input type="submit" value="Save" class="btn btn-default" />
            </div>
        </div>
    </div>
}

<div>
```

```cshtml
@model IEnumerable<MusicStoreApp.Models.Album>

@{
    ViewBag.Title = "Index";
}

<h2>Index</h2>

<p>
    @Html.ActionLink("Create New", "Create")
</p>
<table class="table">
    <tr>
        <th>
            @Html.DisplayNameFor(model => model.Title)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Artist.Name)
        </th>
        <th></th>
    </tr>

@foreach (var item in Model) {
    <tr>
        <td>
            @Html.DisplayFor(modelItem => item.Title)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Artist.Name)
        </td>
        <td>
            @Html.ActionLink("Edit", "Edit", new { id=item.AlbumId }) |
            @Html.ActionLink("Details", "Details", new { id=item.AlbumId }) |
            @Html.ActionLink("Delete", "Delete", new { id=item.AlbumId })
        </td>
    </tr>
}

</table>
```

Τέλος ενώ βρισκόμαστε ακόμα στα Views δημιουργήσαμε συντομεύσεις για τα προαναφερθείσα Views στη κλάση “_Layout.cshtml” του φακέλου Shared ώστε να μπορεί ο χρήστης με ένα κλικ να έχει πρόσβαση στους πίνακες της βάσης.



_Layout.cshtml

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>@ViewBag.Title - My ASP.NET Application</title>
    @Styles.Render("~/Content/css")
    @Scripts.Render("~/bundles/modernizr")
</head>
<body>
    <div class="navbar navbar-inverse navbar-fixed-top">
        <div class="container">
            <div class="navbar-header">
                <button type="button" class="navbar-toggle" data-toggle="collapse" data-target=".navbar-collapse">
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                </button>
                @Html.ActionLink("Chinook Music Store", "Index", "Home", new { area = "" }, new { @class = "navbar-brand" })
            </div>
            <div class="navbar-collapse collapse">
                <ul class="nav navbar-nav">
                    <li>@Html.ActionLink("Home", "Index", "Home")</li>
                    <li>@Html.ActionLink("Artists", "Index", "Artists")</li>
                    <li>@Html.ActionLink("Albums", "Index", "Albums")</li>
                    <li>@Html.ActionLink("Customers", "Index", "Customers")</li>
                    <li>@Html.ActionLink("Employees", "Index", "Employees")</li>
                    <li>@Html.ActionLink("Genres", "Index", "Genres")</li>
                    <li>@Html.ActionLink("Tracks", "Index", "Tracks")</li>
                    <li>@Html.ActionLink("Playlists", "Index", "Playlists")</li>
                    <li>@Html.ActionLink("MediaTypes", "Index", "MediaTypes")</li>
                    <li>@Html.ActionLink("Invoices", "Index", "Invoices")</li>
                    <li>@Html.ActionLink("InvoiceLines", "Index", "InvoiceLines")</li>
                </ul>
            </div>
        </div>
    </div>
    <div class="container body-content">
        @RenderBody()
        <hr />
        <footer>
            <p>&copy; @DateTime.Now.Year - My ASP.NET Application</p>
        </footer>
    </div>
```

100 % No issues found

Το γραφικό περιβάλλον, ο τρόπος χρήσης της εφαρμογής καθώς και παραδείγματα παρουσιάζονται στο Εγχειρίδιο Χρήστη.

Ευχαριστούμε