

۱. 10 آهنگ برتر که بیشترین درآمد رو داشتن به همراه درآمد ایجاد شده

The screenshot shows the SQL Server Enterprise Manager interface. The left pane displays the 'chinook' database schema with tables like album, artist, customer, employee, genre, invoice, invoiceLine, mediatype, playlist, playlisttrack, and track. The central pane shows the following SQL query:

```
1 WITH top10_table AS (  
2   SELECT invl.TrackId, SUM(invl.Quantity * invl.UnitPrice) AS total_sale  
3   FROM invoiceLine invl  
4   GROUP BY invl.TrackId  
5   ORDER BY total_sale DESC  
6   LIMIT 10  
7 )  
8 SELECT t.TrackId, t.Name , invl.total_sale  
9 FROM track t  
10 RIGHT JOIN top10_table invl ON t.TrackId = invl.TrackId;
```

The right pane shows the 'Result Grid' with 13 rows of data. The columns are TrackId, Name, and total\_sale. The data is as follows:

TrackId	Name	total_sale
3200	Gay Witch Hunt	3.98
3250	Pilot	3.98
3214	Phyllis's Wedding	3.98
2832	The Woman King	3.98
2850	The Fix	3.98
3223	How to Stop an Exploding Man	3.98
2868	Walkabout	3.98
3177	Hot Girl	3.98
2833	A Day In the Life	1.99

The bottom pane shows the 'Output' window with the following error message:

```
135 20:09:57 with top10_table as (select invl.TrackId , sum(invl.Quantity*invl.UnitPrice) as total_s... Error Code: 1064. You have an error in your SQL syntax; check th  
136 20:10:58 with top10_table as (select invl.TrackId , sum(invl.Quantity*invl.UnitPrice) as total_s... Error Code: 1054. Unknown column 'invl.TrackId' in 'on clause'
```

SQL script saved to 'C:\Users\orgin\Desktop\DA-Project\Question\1.top 10 sale income track.sql'

۲. محبوب ترین ژانر، به ترتیب از نظر تعداد آهنگهای فروخته شده و کل درآمد

The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'SCHEMAS' pane displays a tree view of the database structure, including 'invoice', 'invoiceLine', and 'Columns'. The 'Columns' pane for 'invoiceLine' lists 'InvoiceLineId' (int PK), 'InvoiceId' (int), 'TrackId' (int), 'UnitPrice' (decimal(10,2)), and 'Quantity' (int). The main pane shows a SQL query in 'Query 1' with the following code:

```
1 WITH sale_table AS (  
2     SELECT invl.TrackId, SUM(invl.Quantity) AS Quantity_sum , sum(invl.UnitPrice) as total_sale  
3     FROM invoiceLine invl  
4     GROUP BY invl.TrackId  
5     order by Quantity_sum desc , total_sale desc  
6 )  
7  
8 SELECT g.Name as genre_name , sum(Quantity_sum) as Quantity , sum(total_sale) as total_sale  
9 FROM track t  
10 JOIN sale_table invl ON t.TrackId = invl.TrackId  
11 join genre g on t.GenreId = g.GenreId  
12 group by t.GenreId  
13 order by Quantity desc , total_sale desc  
14 limit 1  
15 ;  
16
```

Below the query, the 'Result Grid' shows the following data:

genre_name	Quantity	total_sale
Rock	835	826.65

The bottom of the interface shows 'Result 17 x' and 'Output' tabs, with a 'Read Only' status indicator.

۳. کاربرانی که تا حالا خرید نداشتند؟

```
1 select c.CustomerId , inv.InvoiceId from customer c
2 LEFT OUTER JOIN invoice inv on c.CustomerId = inv.CustomerId
3 where inv.InvoiceId is null ;
```

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
CustomerId	InvoiceId		

تمامی کاربران موجود در تیبل customer خرید داشته اند.


برای مطمئن شدن این موضوع می توانیم تعداد مشتری های یکتا دوتا جدول رو مقایسه کنیم (customer) invoice and

2. Popular genre*	SQL File 3*	SQL File 4*
1 • select count(c.CustomerId) from customer c		
Result Grid		
count(c.CustomerId)		
59		




2. Popular genre*	SQL File 3*	SQL File 4*
1 • select count(DISTINCT inv.CustomerId) from invoice inv		
2		
Result Grid		
count(DISTINCT inv.CustomerId)		
59		

۴. میانگین زمان آهنگ ها در هر آلبوم (به ترتیب نزولی میانگین زمان (میلی ثانیه))

3.Customer\_with\_no\_sale\* x SQL File 3\* SQL File 4\*

 Don't Limit

```
1 select a.AlbumId , a.Title , avg(t.Milliseconds) as track_time from track t
2 join album a on t.AlbumId = a.AlbumId
3 group by a.AlbumId , a.Title
4 order by track_time desc;
```

Result Grid |  Filter Rows: | Export:  Wrap Cell Content: 

	AlbumId	Title	track_time
▶	253	Battlestar Galactica (Classic), Season 1	2925574.3333
	227	Battlestar Galactica, Season 3	2778265.3158
	229	Lost, Season 3	2717907.0000
	231	Lost, Season 2	2637067.9583
	226	Battlestar Galactica: The Story So Far	2622250.0000
	228	Heroes, Season 1	2599142.0870
	230	Lost, Season 1	2594197.4400
	254	Aquaman	2484567.0000
	261	LOST, Season 4	2321672.5294
	251	The Office, Season 3	1532683.8000
	249	The Office, Season 1	1329194.0000
	250	The Office, Season 2	1301645.7273
	50	The Final Concerts (Disc 2)	932094.0000
	138	The Song Remains The Same (Disc 2)	759359.7500
	198	Santana Live	673923.8333
	294	Great Performances - Barber's Adagio ...	596519.0000
	137	The Song Remains The Same (Disc 1)	588794.2000
	279	Handel: The Messiah (Highlights)	582029.0000
	208	[1997] Black Light Syndrome	575790.5714
	330	Górecki: Symphony No. 3	567494.0000
	212	...	561067.0000

Result 26 x

Output

۵) کارمندی که بیشترین تعداد فروش را داشته

The screenshot shows a SQL IDE interface with a query editor and a results grid. The query editor contains the following SQL code:

```
1 select e.EmployeeId , e.FirstName , e.LastName , e.Title , e.Country , e.Phone
2     , sum(invl.Quantity) Quantity_sale from employee e
3 left join customer c on e.EmployeeId = c.SupportRepId
4 join invoice inv on c.CustomerId = inv.CustomerId
5 join invoiceline invl on inv.InvoiceId = invl.InvoiceId
6 group by e.EmployeeId
7 limit 1;
```

The results grid displays the following data:

EmployeeId	FirstName	LastName	Title	Country	Phone	Quantity_sale
3	Jane	Peacock	Sales Support Agent	Canada	+1 (403) 262-3443	796

Below the results grid, there is a section labeled "Result 32" and an "Output" area.

۶) کاربرانی که از بیش از یک ژانر خرید کردند

5.Top\_employee\_sale\_quantity\* x SQL File 3\* SQL File 4\*

Don't Limit

```
1 • select c.CustomerId , count(distinct g.GenreId) genre_count from customer c
2 left join invoice inv on c.CustomerId = inv.CustomerId
3 join invoiceline invl on inv.InvoiceId = invl.InvoiceId
4 join track t on t.TrackId = invl.TrackId
5 join genre g on g.GenreId = t.GenreId
6 group by c.CustomerId
7 HAVING COUNT(DISTINCT g.GenreId) > 1;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

	CustomerId	genre_count
▶	1	8
	2	7
	3	10
	4	8
	5	8
	6	9
	7	9
	8	4
	9	5
	10	7
	11	6
	12	5
	13	7
	14	10
	15	8
	16	7
	17	10
	18	6

Result 36 x Read Only

Output

Result Grid  
Form Editor  
Field Types  
Query Stats  
Execution Plan

۷) سه آهنگ برتر از نظر درآمد فروش برای هر ژانر

5.Top\_employee\_sale\_quantity\* x SQL File 3\* SQL File 4\*

```
1 WITH sale_table AS (
2     SELECT invl.TrackId, SUM(invl.Quantity*invl.UnitPrice) as total_sale
3     FROM invoiceline invl
4     GROUP BY invl.TrackId
5 ),
6 ranked_sales as (
7     SELECT g.Name as genre_name , g.GenreId, invl.total_sale
8     , row_number() over (partition by g.GenreId order by total_sale desc) as row_id
9     FROM track t
10    JOIN sale_table invl ON t.TrackId = invl.TrackId
11    join genre g on t.GenreId = g.GenreId
12 )
13 SELECT genre_name , GenreId, total_sale
14 FROM ranked_sales
15 WHERE row_id <= 3;
16
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

genre_name	GenreId	total_sale
Rock	1	1.98
Rock	1	1.98
Rock	1	1.98
Jazz	2	1.98
Jazz	2	1.98
Jazz	2	1.98
Metal	3	1.98

Result 43 x Read Only

5.Top\_employee\_sale\_quantity\* x SQL File 3\* SQL File 4\*

```
1 WITH sale_table AS (
2     SELECT invl.TrackId, SUM(invl.Quantity*invl.UnitPrice) as total_sale
3     FROM invoiceline invl
4     GROUP BY invl.TrackId
5 ),
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

genre_name	GenreId	total_sale
Rock	1	1.98
Rock	1	1.98
Rock	1	1.98
Jazz	2	1.98
Jazz	2	1.98
Jazz	2	1.98
Metal	3	1.98
Metal	3	1.98
Metal	3	1.98
Alternative & Punk	4	1.98
Alternative & Punk	4	1.98
Alternative & Punk	4	1.98
Rock And Roll	5	0.99
Rock And Roll	5	0.99
Rock And Roll	5	0.99
Blues	6	1.98
Blues	6	1.98
Blues	6	1.98
Latin	7	1.98
Latin	7	1.98

Result 43 x Read Only

Output

۸) تعداد آهنگهای فروخته شده به صورت تجمعی در هر سال به صورت جداگانه

6.customer\_having\_2\_or\_more\_... x SQL File 3\* SQL File 4\*

1 • select year(inv.InvoiceDate) as year ,  
2 sum(inv1.Quantity) over ( partition by year(inv.InvoiceDate) order by inv.InvoiceId ) from invoice inv  
3 join invoice1 inv1 on inv1.InvoiceId = inv.InvoiceId  
4 )

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows: |

	year	sum(inv1.Quantity) over ( partition by year(inv.InvoiceDate) order by inv.InvoiceId )
▶	2021	2
	2021	2
	2021	6
	2021	6
	2021	6
	2021	12
	2021	12
	2021	12
	2021	12
	2021	12
	2021	12
	2021	21
	2021	21

Result 54 x Read Only

Output

Action Output

#	Time	Action	Message
80	18:51:12	select * from (select year(inv.InvoiceDate) as year , sum(inv1.Quantity) over ( partition by year(inv.InvoiceDate) order by inv.InvoiceId ) from invoice inv join invoice1 inv1 on inv1.InvoiceId = inv.InvoiceId )	Error Code: 1248. Every derived table must have its own alias.
81	18:51:24	select year(inv.InvoiceDate) as year , sum(inv1.Quantity) over ( partition by year(inv.InvoiceDate) order by inv.InvoiceId ) from invoice inv join invoice1 inv1 on inv1.InvoiceId = inv.InvoiceId	2240 row(s) returned

Windows taskbar: File Explorer, Google Chrome, Telegram, Signal, N, W, P, Task Manager, ENG



۹) کاربرانی که مجموع خریدشان بالاتر از میانگین مجموع خرید تمام کاربران است.

8.each\_year\_sale\_quantity\* x SQL File 3\* SQL File 4\*

Don't Limit

```
1 WITH total_sale AS (  
2     SELECT invl.TrackId, SUM(invl.Quantity * invl.UnitPrice) AS total_sale  
3     FROM invoiceline invl  
4     GROUP BY invl.TrackId  
5     ORDER BY total_sale DESC  
6 )  
7 select c.CustomerId , sum(total_sale) C_sale from customer c  
8 left join invoice inv on c.CustomerId = inv.CustomerId  
9 join invoiceline invl on inv.InvoiceId = invl.InvoiceId  
10 join total_sale ts on ts.TrackId = invl.TrackId  
11 GROUP BY c.CustomerId  
12 having C_sale > (  
13     select avg(TotalPurchase) from (  
14         select c.CustomerId ,  
15             sum(total_sale) as TotalPurchase from customer c  
16         left join invoice inv on c.CustomerId = inv.CustomerId  
17         join invoiceline invl on inv.InvoiceId = invl.InvoiceId  
18         join total_sale ts on ts.TrackId = invl.TrackId  
19         GROUP BY c.CustomerId)  
20     ) as Subquery  
21 ) ;  
22  
23
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

	CustomerId	C_sale
	1	50.51
	4	49.52
	5	50.53
	6	60.54
	7	49.55

Result 70 x

Read Only

Output :

8.each\_year\_sale\_quantity\* x SQL File 3\* SQL File 4\*

Don't Limit

```

1 WITH total_sale AS (
2     SELECT invl.TrackId, SUM(invl.Quantity * invl.UnitPrice) AS total_sale
3     FROM invoiceline invl
4     GROUP BY invl.TrackId
5     ORDER BY total_sale DESC
6 )
7 select c.CustomerId, sum(total_sale) C_sale from customer c

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

CustomerId	C_sale
1	50.51
4	49.52
5	50.53
6	60.54
7	49.55
8	49.50
17	48.53
18	48.51
19	49.52
22	50.51
24	49.57
25	54.51
26	56.53
28	50.55
34	49.53
35	48.51
37	49.56
39	50.51
40	48.52
42	48.53

Result 70 x

Output