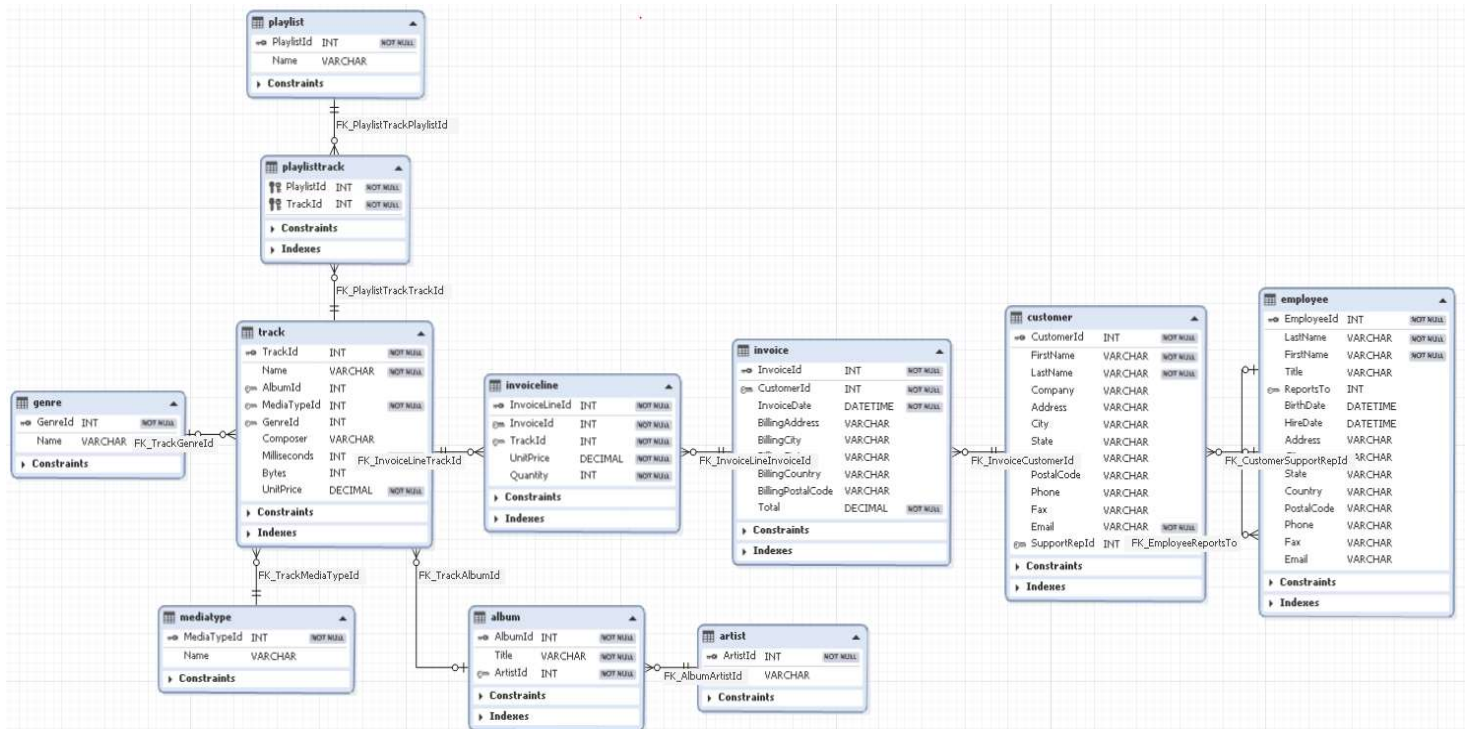


Chin00k project sql

Provided by Mahdiyeh Alavi

CHINOOK DATA SQL

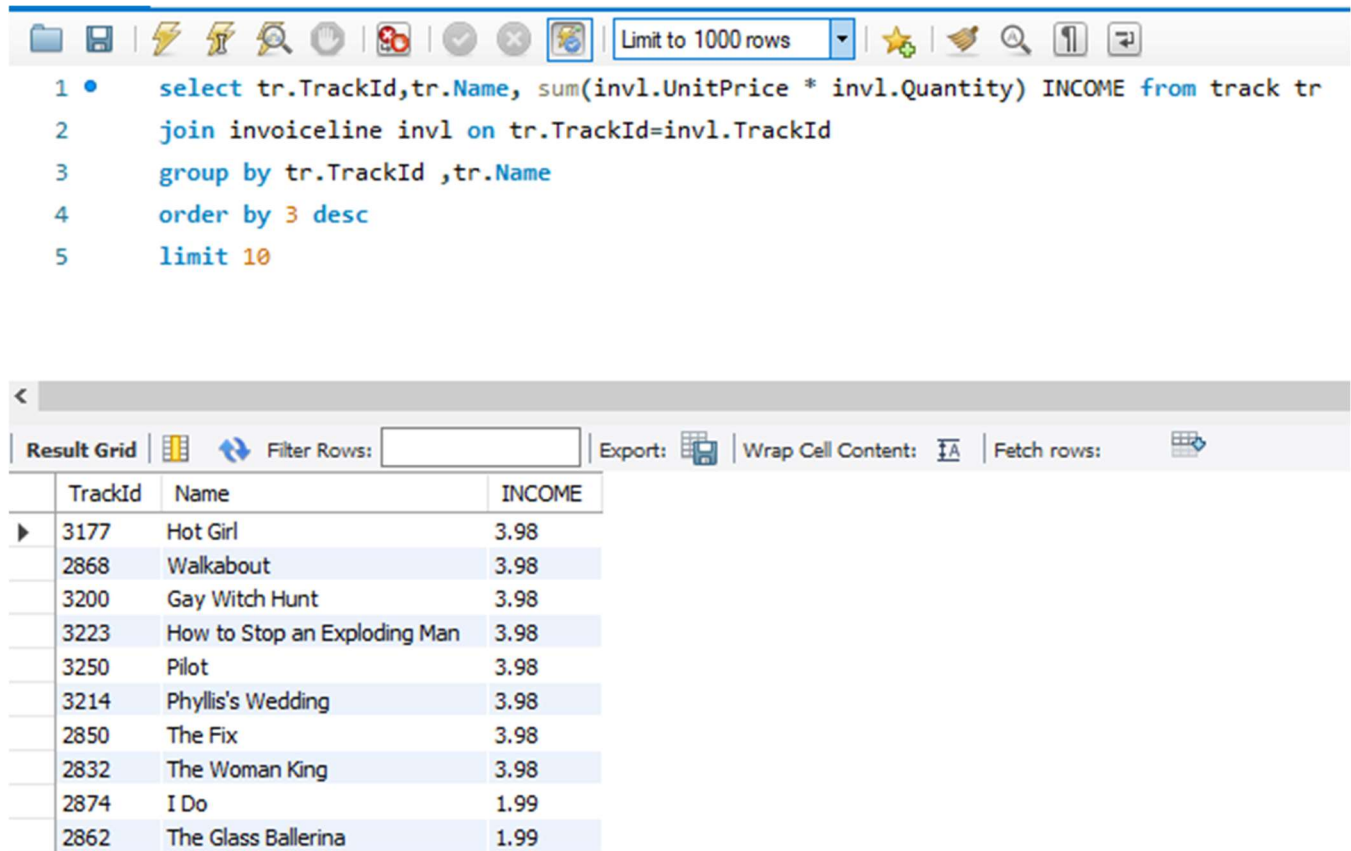
DIAGRAM:



The **InvoiceLine** table includes quantity and price of tracks so it can be the **FACT** type and The rest of the tables include the information and description of the parameters in this table ,so they are **DIM** tables. Because we have only one fact table and also some of DIM tables have branches, this diagram is similar to **snowflake** type.

CHINOOK DATA SQL

1) The top 10 songs that have the most income are created along with the income:



The screenshot shows a SQL query editor with a toolbar at the top. The query is as follows:

```
1 • select tr.TrackId, tr.Name, sum(invl.UnitPrice * invl.Quantity) INCOME from track tr
2 join invoice line invl on tr.TrackId=invl.TrackId
3 group by tr.TrackId ,tr.Name
4 order by 3 desc
5 limit 10
```

Below the query editor, the results are displayed in a table with columns: TrackId, Name, and INCOME. The table contains 10 rows of data, with the first 9 rows having an income of 3.98 and the last row having an income of 1.99.

TrackId	Name	INCOME
3177	Hot Girl	3.98
2868	Walkabout	3.98
3200	Gay Witch Hunt	3.98
3223	How to Stop an Exploding Man	3.98
3250	Pilot	3.98
3214	Phyllis's Wedding	3.98
2850	The Fix	3.98
2832	The Woman King	3.98
2874	I Do	1.99
2862	The Glass Ballerina	1.99

2) the most popular genre, in terms of number of songs sold and Income, respectively:

```
1
2 • select tr.GenreId,gn.Name, count(invl.Quantity)TOTALSOLD,sum(invl.UnitPrice * invl.Quantity) INCOME from track tr
3   join invoiceline invl on tr.TrackId=invl.TrackId
4   join genre gn on tr.GenreId=gn.GenreId
5   group by tr.GenreId
6   order by 3 desc , 4 desc
7   limit 1
```

<

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

	GenreId	Name	TOTALSOLD	INCOME
▶	1	Rock	835	826.65

3) Users who have not purchased before:

```

1 • select cust.CustomerId ,sum(inv.Total) TOTALSOLD from customer cust
2 left join invoice inv on inv.CustomerId=cust.CustomerId
3 group by cust.CustomerId
4 having TOTALSOLD=0

```

Result Grid			Filter Rows: <input type="text"/>	Export:	Wrap Cell Content:
CustomerId	TOTALSOLD				


4) Average time of songs in each album


```

1 • select tr.albumid ,al.title ,avg(tr.milliseconds) Average_durationtime_of_track from track tr
2 join album al on al.albumid=tr.albumid
3 group by tr.albumid,al.title

```


Result Grid






Filter Rows:

Export:



Wrap Cell Content:



	albumid	title	Average_durationtime_of_track
▶	1	For Those About To Rock We Salute You	240041.5000
	2	Balls to the Wall	342562.0000
	3	Restless and Wild	286029.3333
	4	Let There Be Rock	306657.3750
	5	Big Ones	294113.9333
	6	Jagged Little Pill	265455.7692
	7	Facelift	270780.4167
	8	Warner 25 Anos	207637.5714
	9	Plays Metallica By Four Cellos	333925.8750

5) The employee who had the highest number of sales :

```

1 • select emp.employeeid,emp.firstname ,emp.lastname,count(invl.quantity) quantity_sold from employee emp
2   join customer cust on emp.employeeid=cust.supportrepid
3   join invoice inv on inv.customerid=cust.customerid
4   join invoiceline invl on invl.invoiceid=inv.invoiceid
5   group by emp.employeeid,emp.firstname ,emp.lastname
6   order by 4 desc
7   limit 1

```

result Grid	Filter Rows:	Export:	Wrap Cell Content:	Fetch rows:
employeeid	firstname	lastname	quantity_sold	
3	Jane	Peacock	796	

6) Users who buy from more than one genre:

```

1 • select cust.CustomerId,cust.FirstName,cust.LastName,count(distinct gn.genreid) total_gener_bought from customer cust
2   join invoice inv on inv.CustomerId=cust.CustomerId
3   join invoiceline invl on inv.invoiceid=invl.invoiceid
4   join track tr on tr.TrackId=invl.TrackId
5   join genre gn on tr.GenreId=gn.GenreId
6   group by cust.CustomerId,cust.FirstName,cust.LastName
7   having total_gener_bought >1
8

```

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
CustomerId	FirstName	LastName	total_gener_bought
1	Luis	Gonçalves	8
2	Leonie	Köhler	7
3	François	Tremblay	10
4	Björn	Hansen	8
5	František	Wichterlová	8
6	Helena	Holý	9
7	Astrid	Gruber	9
8	Daan	Peeters	4
9	Kara	Nielsen	5
10	Eduardo	Martins	7
11	Alexandre	Rocha	6
12	Roberto	Almeida	5
13	Fernanda	Ramos	7
..

7) the top three songs in terms of sales revenue for each genre :

```

1 • with TRACK_INCOME as
2 (
3   select tr.TrackId,gn.Name as Genre,tr.name as Track ,SUM(invl.Quantity * invl.UnitPrice) Income from track tr
4     join invoiceline invl on tr.TrackId = invl.TrackId
5     join genre gn on tr.GenreId = gn.GenreId
6   GROUP BY Genre, tr.TrackId ,Track
7 ),
8   SORTED_TRACK_INCOME as
9 (
10  select Genre,Track,Income,ROW_NUMBER() OVER (PARTITION BY Genre ORDER BY Income desc) row_num from TRACK_INCOME
11  ORDER BY Genre, Income desc
12 )
13  select Genre,Track,Income from SORTED_TRACK_INCOME
14  WHERE row_num <= 3
15

```

Result Grid			
Filter Rows:		Export:	Wrap Cell Content: IA
Genre	Track	Income	
Alternative	Show Me How to Live (Live at the Quart Festival)	0.99	
Alternative	Say Hello 2 Heaven	0.99	
Alternative	Until We Fall	0.99	
Alternative & Punk	Waiting	1.98	
Alternative & Punk	Real Love	1.98	
Alternative & Punk	I Believe	1.98	
Blues	Promises	1.98	
Blues	Sunshine Of Your Love	1.98	
Blues	Lay Down Sally	1.98	
Bossa Nova	Onde Anda Você	1.98	
Bossa Nova	Formosa	0.99	
Bossa Nova	Deixa	0.99	

8) the number of songs sold cumulatively in each year

```

1 • with total_quantity_invoice_date as
2 (
3   select inv.InvoiceId, inv.InvoiceDate, sum(invL.Quantity) total_quantity, extract(YEAR FROM inv.InvoiceDate) invyear
4   from invoice inv
5   join invoiceline invl on inv.InvoiceId=invl.InvoiceId
6   group by inv.InvoiceId, inv.InvoiceDate
7 )
8 select tqid.InvoiceId, tqid.InvoiceDate, tqid.total_quantity,
9        sum(tqid.total_quantity) over(partition by tqid.invyear order by tqid.InvoiceId) collective_quantity
10 from total_quantity_invoice_date tqid

```

Set limit for number of rows returned by queries.
Workbench will automatically add the LIMIT clause with the configured number of rows to SELECT queries.

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



	InvoiceId	InvoiceDate	total_quantity	collective_quantity
70	2021-11-07 00:00:00	2	380	
71	2021-11-07 00:00:00	2	382	
72	2021-11-08 00:00:00	4	386	
73	2021-11-09 00:00:00	6	392	
74	2021-11-12 00:00:00	9	401	
75	2021-11-17 00:00:00	14	415	
76	2021-11-25 00:00:00	1	416	
77	2021-12-08 00:00:00	2	418	
78	2021-12-08 00:00:00	2	420	
79	2021-12-09 00:00:00	4	424	
80	2021-12-10 00:00:00	6	430	
81	2021-12-13 00:00:00	9	439	
82	2021-12-18 00:00:00	14	453	
83	2021-12-26 00:00:00	1	454	
84	2022-01-08 00:00:00	2	2	
85	2022-01-08 00:00:00	2	4	
86	2022-01-09 00:00:00	4	8	
87	2022-01-10 00:00:00	6	14	
88	2022-01-13 00:00:00	9	23	
89	2022-01-18 00:00:00	14	37	
90	2022-01-26 00:00:00	1	38	
91	2022-02-08 00:00:00	2	40	
92	2022-02-08 00:00:00	2	42	
93	2022-02-09 00:00:00	4	46	
94	2022-02-10 00:00:00	6	52	

9) users whose total purchase is higher than the average total purchase of all users :

```

1 • with sum_total_bought as
2   (
3     select inv.CustomerId,sum(inv.Total)sum_total
4     from invoice inv
5
6     group by inv.CustomerId
7
8   ),
9   avgtotal as
10  (
11    select avg(stb.sum_total)avg_total from sum_total_bought stb
12
13  )
14  select stb.customerid,cust.FirstName,cust.LastName,stb.sum_total ,(select avgt.avg_total from avgtotal avgt)average_total
15  from sum_total_bought stb
16  join customer cust on cust.CustomerId=stb.customerid
17  having sum_total>average_total

```

Result Grid					
Filter Rows:		Export:  Wrap Cell Content: 			
	customerid	FirstName	LastName	sum_total	average_total
▶	1	Luis	Gonçalves	39.62	39.467797
	3	François	Tremblay	39.62	39.467797
	4	Björn	Hansen	39.62	39.467797
	5	František	Wichterlová	40.62	39.467797
	6	Helena	Holý	49.62	39.467797
	7	Astrid	Gruber	42.62	39.467797
	17	Jack	Smith	39.62	39.467797
	20	Dan	Miller	39.62	39.467797
	22	Heather	Leacock	39.62	39.467797
	24	Frank	Ralston	43.62	39.467797
	25	Victor	Stevens	42.62	39.467797
	26	Richard	Cunningham	47.62	39.467797
	28	Julia	Barnett	43.62	39.467797
	34	João	Fernandes	39.62	39.467797
	37	Fynn	Zimmermann	43.62	39.467797
	42	Wyatt	Girard	39.62	39.467797
	43	Isabelle	Mercier	40.62	39.467797
	44	Terhi	Hämäläinen	41.62	39.467797
	45	Ladislav	Kovács	45.62	39.467797
	46	Hugh	O'Reilly	45.62	39.467797
	48	Johannes	Van der Berg	40.62	39.467797
	57	Luis	Rojas	46.62	39.467797