DBMS Project Report

PES University

Database Management Systems

UE18CS252

Submitted By

PES2201800064 SKANDAN K A

SUMMARY OF THE PROJECT:

Publishers database is a database which manages all the activities carried on from publishing house to the shops, when a new book is published.This database is created using sql server.
It uses several tables like sales,titles,authors etc.
Detailed data model along with the datatypes and keys, is described in the data model section.
The er diagram consisting of all the tables and the schema diagram is shown.
Table creation and usage of ddl commands along with the screenshots of the sql server can be seen.
Normalization and functional dependencies along with the reason for the normal form is also written.
A Trigger called Tr_admin is created to make sure that a table named admin cannot be deleted and it displays a print statement that this table cannot be deleted. However if the trigger is disabled, it can be deleted.
2 correlated subqueries are shown to extract number of employees for a particular designation and get the max job_level among the employees with same id respectively.
A Full outer join and a right outer join is done on titles and publishers tables.
Finally a brief description regarding advantages, limitations and enhancements in the conclusion section.

Introduction

Database used- Publishers Database (Pubs Database)
The PUBS database includes a fictional set of information about

1. publishers,
2. authors,
3. titles and
4. the sales of their associated books.

The database is often used as a model database that can be experimented with.
It has well defined data about the publishing house.
This database can manage all the details with respect to published books and their authors ,its sales,
about the publishers,the type of jobs and range of employees in the publishing house,quantity of books ordered and about the repective employees involved in this process.
Thus it is highly helpful in this entire journey from publishing house to the customer.
This database has tables like authors,discounts,employee,jobs,pub_info etc..
It is executed in Microsoft Sql Server.
Sales table has attributes like stor_id,ord_num,title_id,qty,payterms.
Stores table has attributes like stor_id,stor_name,stor_address,
Discounts table has attributes like stor_id,lowqty,highqty
Roysched table has information about royalty of the book published.
Pub_info table has columns like pub_id,logo,pr_info..
The constituents and creation of all the tables is shown in the upcoming pages, to make the report more qualitative.
Transaction is used in jobs table to find the min and max level of jobs for a particular job.
1st job is given by default level of 10.All the other job types can have the required levels.

Data Model of the Tables

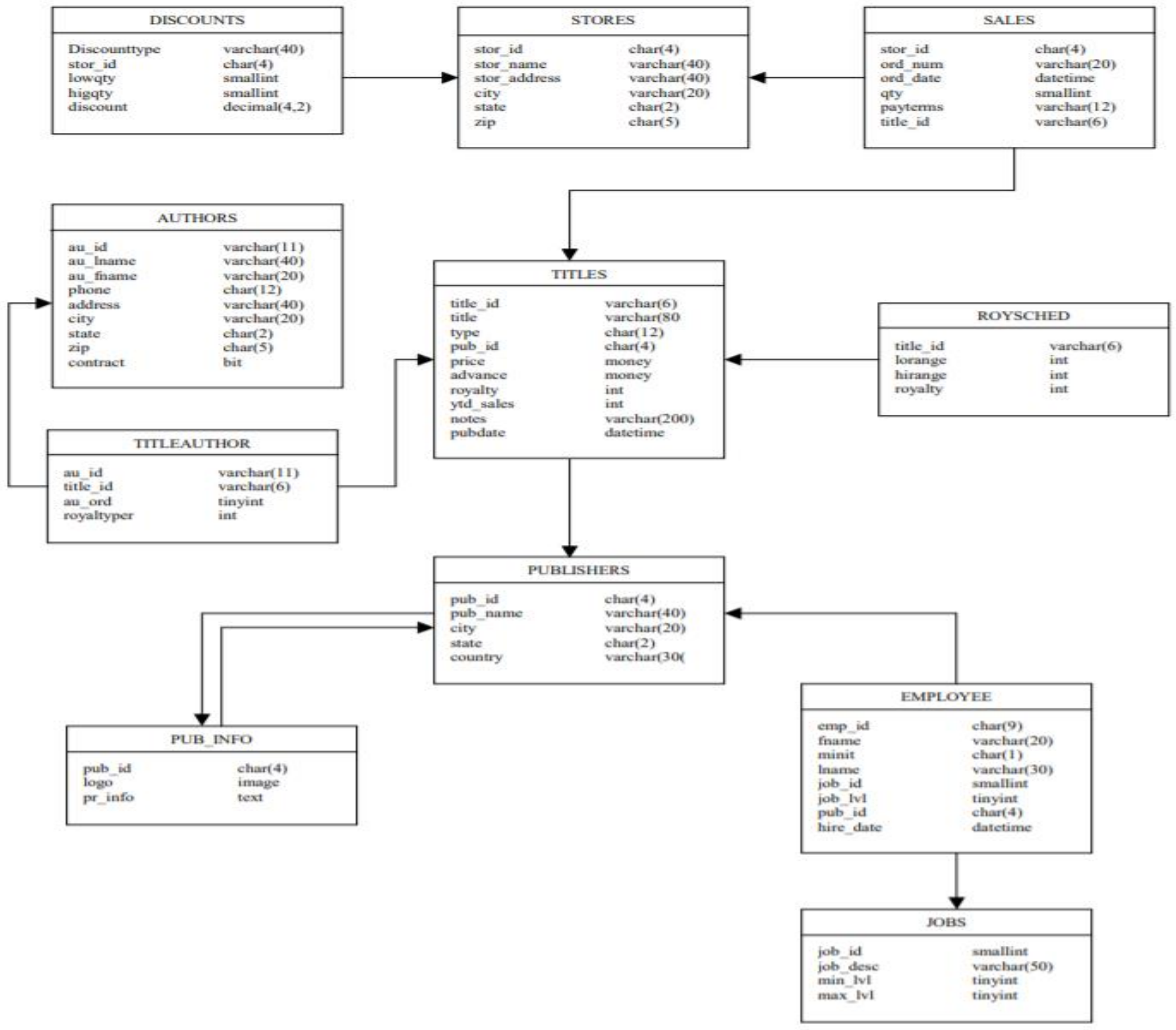
```
table(dbo.pub_info) {
        pub_id: char <<PK>>
        pub_id: char <<FK>>
        pub_id: char
        logo: image
        pr_info: text
```

```
          }


table(dbo.sales) {
          stor_id: char <<PK>>
          ord_num: varchar <<PK>>
          title_id: varchar <<PK>>
          stor_id: char <<FK>>
          title_id: varchar <<FK>>
          stor_id: char
          ord_num: varchar
          ord_date: datetime
          qty: smallint
          payterms: varchar
          title_id: varchar
}


table(dbo.stores) {
          stor_id: char <<PK>>
          stor_id: char
          stor_name: varchar
          stor_address: varchar
          city: varchar
          state: char
          zip: char
}




table(dbo.discounts) {
          stor_id: char <<FK>>
          discounttype: varchar
          stor_id: char
          lowqty: smallint
          highqty: smallint
          discount: decimal
}


table(dbo.authors) {
          au_id: varchar <<PK>>
          au_id: varchar
          au_lname: varchar
          au_fname: varchar
          phone: char
          address: varchar
          city: varchar
          state: char
          zip: char
          contract: bit
}
```

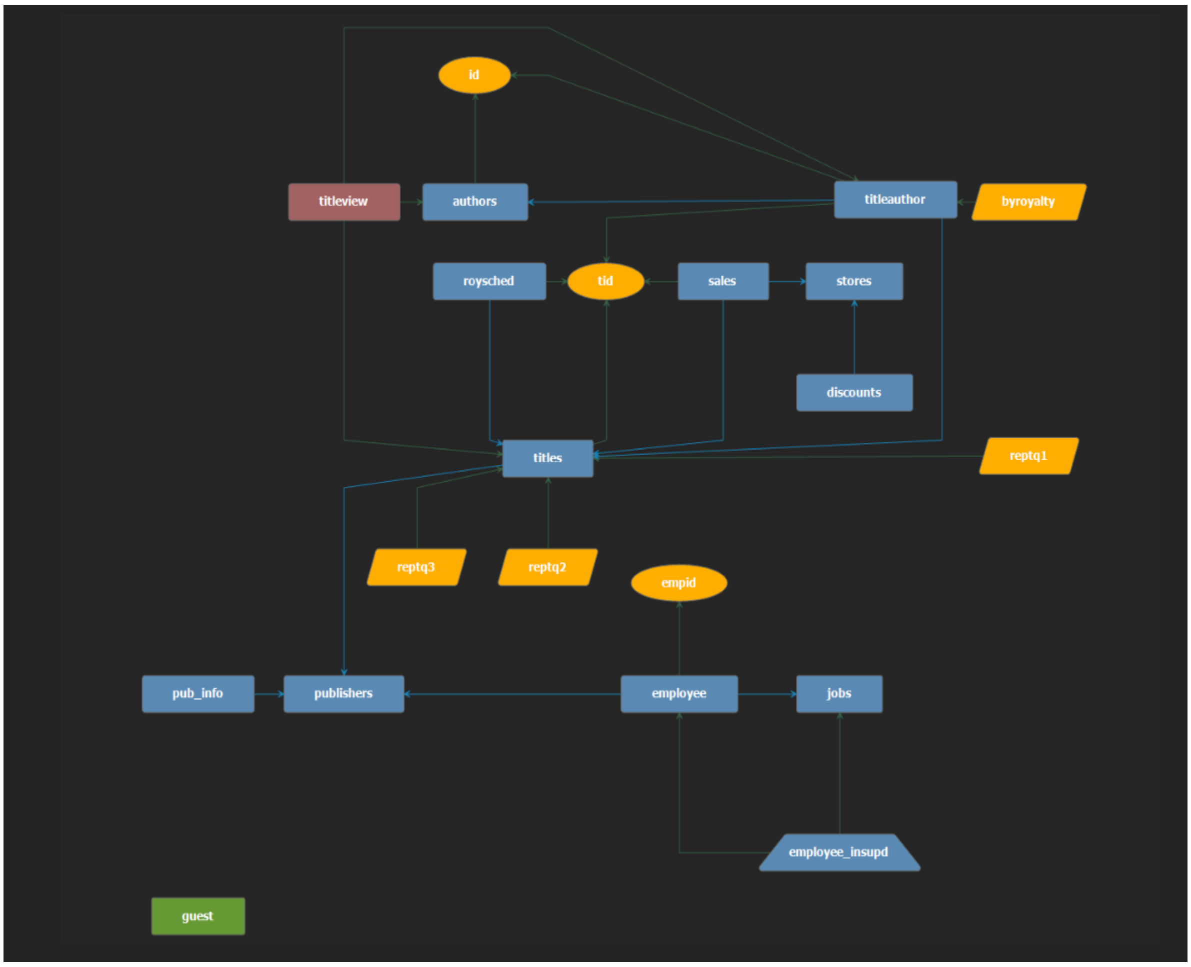
```
table(dbo.roysched) {
        title_id: varchar <<FK>>
        title_id: varchar
        lorange: int
        hirange: int
        royalty: int
}


table(dbo.titleauthor) {
        au_id: varchar <<PK>>
        title_id: varchar <<PK>>
        au_id: varchar <<FK>>
        title_id: varchar <<FK>>
        au_id: varchar
        title_id: varchar
        au_ord: tinyint
        royaltyper: int
}

table(dbo.employee) {

        emp_id: char <<PK>>

        job_id: smallint <<FK>>

        pub_id: char <<FK>>

        emp_id: char

        fname: varchar

        minit: char

        lname: varchar

        job_id: smallint

        job_lvl: tinyint

        pub_id: char

        hire_date: datetime
}
```

KEYS USED:

dbo.discounts -|> dbo.stores:FK
dbo.employee -|> dbo.jobs:FK
dbo.employee -|> dbo.publishers:FK
dbo.pub_info -|> dbo.publishers:FK
dbo.roysched -|> dbo.titles:FK
dbo.sales -|> dbo.stores:FK
dbo.sales -|> dbo.titles:FK
dbo.titleauthor -|> dbo.authors:FK
dbo.titleauthor -|> dbo.titles:FK
dbo.titles -|> dbo.publishers:FK



SCHEMA REPRESENTATION



E R DIAGRAM

FD and Normalization

Functional dependencies of the relations:
In sales, {title_id}->{stor_id,ord_date,ord_num}
In pub_info, {pub_id}->{pr_info,logo}
In stores, {store_id}->{store_name,store_add}
{store_address,zip}->{city}
In jobs, {job_id}->{job_desc,min_lvl,max_lvl}
{job_desc}->{job_id,min_lvl,max_lvl}
In titles, {title_id}->{title,type,price,advance}
In employee, {emp_id}->{fname,lname,job_lvl}
In titleauthor, {title_id}->{au_order,au_id}

Normalization:
'Jobs' is in BCNF. If non-key attribute man_lvl could find another attribute, say no_mangr which could inturn be found out by job_id/job_desc causes a transitive dependency thus violating BCNF and 3NF also.
'Sales' is in 2NF as ord_num(NON-key attribute) can find ord_date(another non-key attribute),thus violating 3NF.

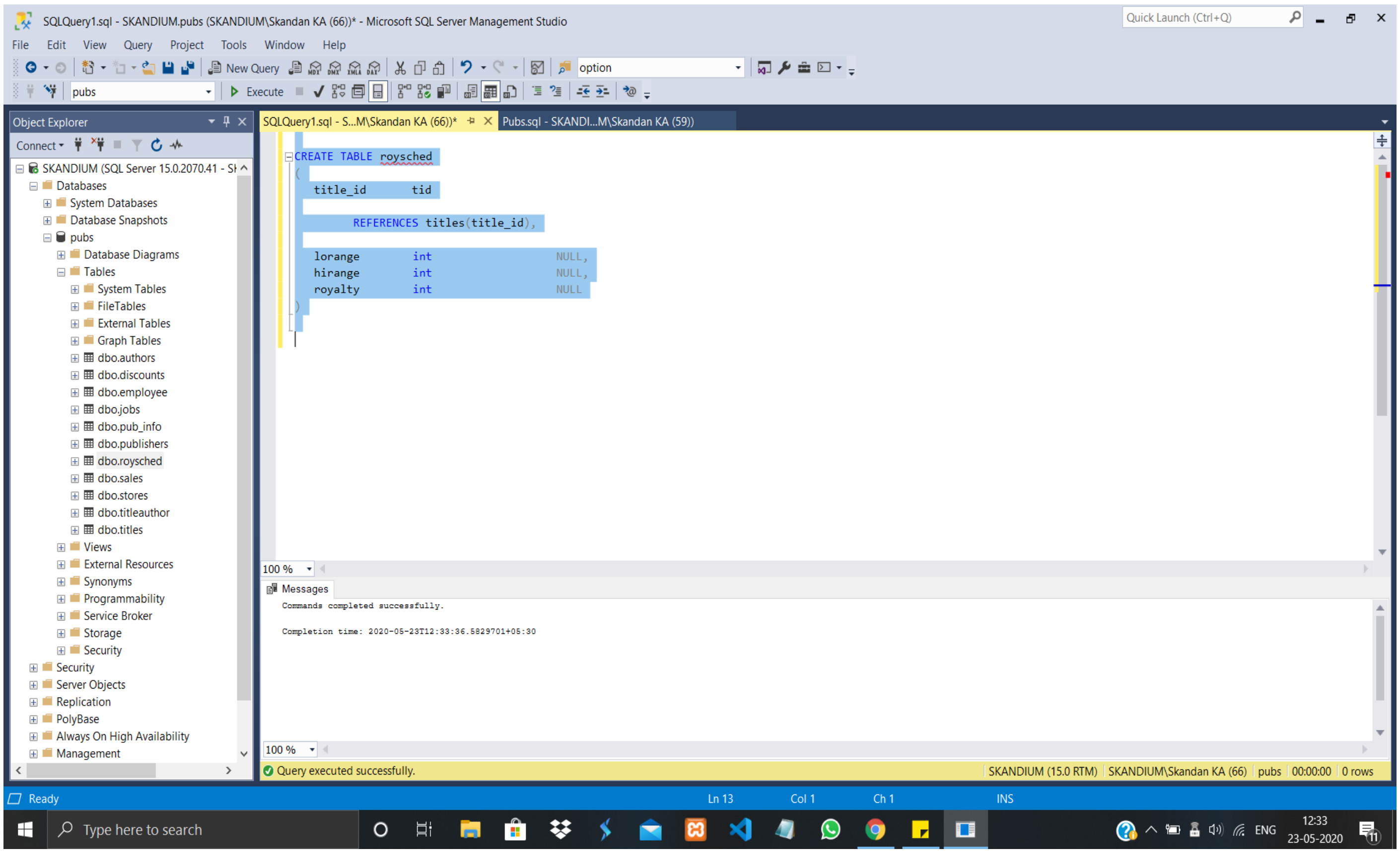
‘Pub_info’ is in BCNF. If another non key attribute publisher_name, which could find pr_info was added(not a superkey), it would have violated the BCNF.
‘Stores’ is in 3NF and not in BCNF as stor_id(key attribute) can derive another key attribute(store_name,store_address).
‘Employee’ is in BCNF as there are no partial or transitive dependencies.
‘Titles’ is in BCNF.
‘Titleauthor’ is in BCNF.

DDL

```
CREATE TABLE pub_info
(
   pub_id         char(4)           NOT NULL

         REFERENCES publishers(pub_id)

         CONSTRAINT UPKCL_pubinfo PRIMARY KEY CLUSTERED,

   logo           image                 NULL,
   pr_info        text                  NULL
)


CREATE TABLE discounts
(
   discounttype   varchar(40)       NOT NULL,

   stor_id        char(4) NULL

         REFERENCES stores(stor_id),

   lowqty         smallint              NULL,
   highqty        smallint              NULL,
   discount       dec(4,2)          NOT NULL
)


CREATE TABLE stores
(
   stor_id        char(4)           NOT NULL

         CONSTRAINT UPK_storeid PRIMARY KEY CLUSTERED,

   stor_name      varchar(40)           NULL,
   stor_address   varchar(40)           NULL,
   city           varchar(20)           NULL,
   state          char(2)               NULL,
   zip            char(5)               NULL
)


CREATE TABLE sales
(
   stor_id        char(4)           NOT NULL

         REFERENCES stores(stor_id),

   ord_num        varchar(20)       NOT NULL,
   ord_date       datetime          NOT NULL,
   qty            smallint          NOT NULL,
   payterms       varchar(12)       NOT NULL,

   title_id       tid
```

```
CREATE TABLE titles
(
   title_id       tid

         CONSTRAINT UPKCL_titleidind PRIMARY KEY CLUSTERED,

   title          varchar(80)       NOT NULL,

   type           char(12)          NOT NULL

         DEFAULT ('UNDECIDED'),

   pub_id         char(4)               NULL

         REFERENCES publishers(pub_id),

   price          money                 NULL,
   advance        money                 NULL,
   royalty        int                   NULL,
   ytd_sales      int                   NULL,
   notes          varchar(200)          NULL,

   pubdate        datetime          NOT NULL

         DEFAULT (getdate())
)



CREATE TABLE titleauthor
(
   au_id          id

         REFERENCES authors(au_id),

   title_id       tid

         REFERENCES titles(title_id),

   au_ord         tinyint               NULL,
   royaltyper     int                   NULL,


   CONSTRAINT UPKCL_taind PRIMARY KEY CLUSTERED(au_id, title_id)
)


CREATE TABLE roysched
(
   title_id       tid

         REFERENCES titles(title_id),

   lorange        int                   NULL,
   hirange        int                   NULL,
   royalty        int                   NULL
)



CREATE TABLE jobs
(
   job_id         smallint          IDENTITY(1,1)

         PRIMARY KEY CLUSTERED,
   job_desc       varchar(50)       NOT NULL

         DEFAULT 'New Position - title not formalized yet',

   min_lvl        tinyint           NOT NULL

         CHECK (min_lvl >= 10),

   max_lvl        tinyint           NOT NULL

         CHECK (max_lvl <= 250)
)
```

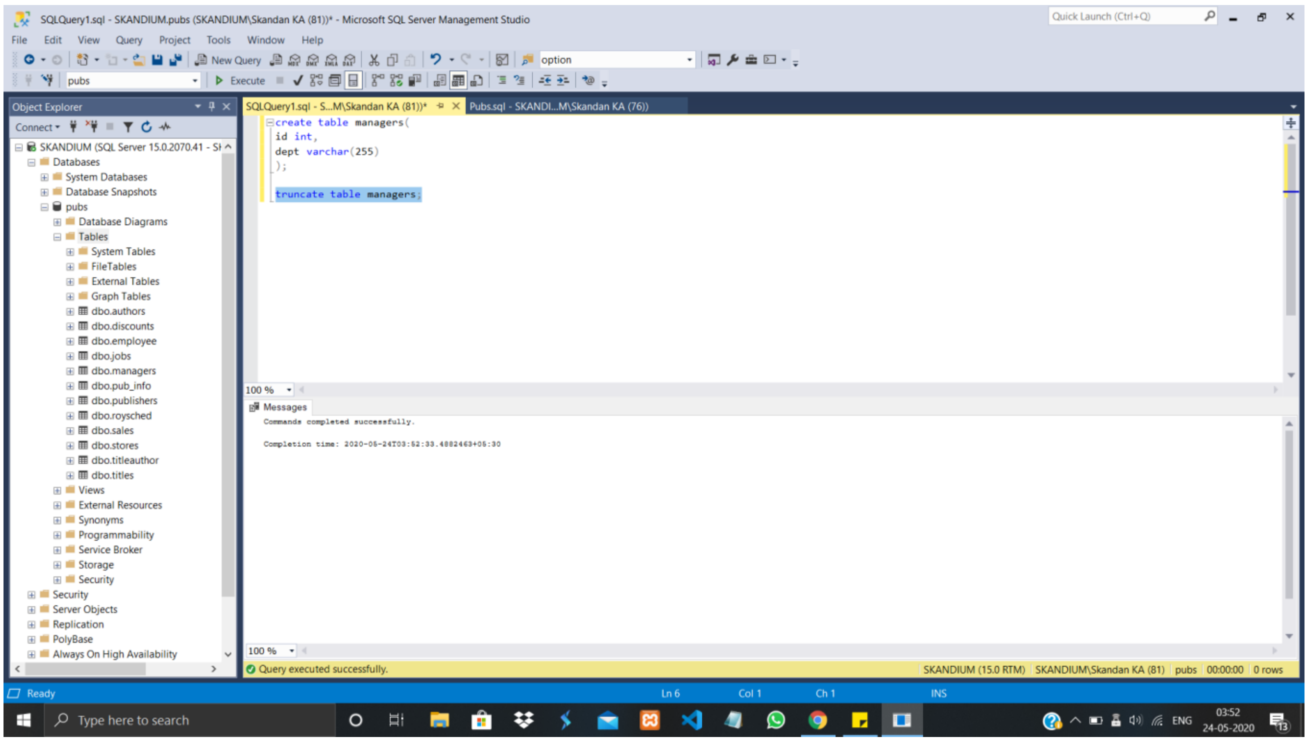
Creation of ROYSCHED table in Sql Server



```
CREATE TABLE roysched
(
  title_id       tid

         REFERENCES titles(title_id),

  lorange        int                   NULL,
  hirange        int                   NULL,
  royalty        int                   NULL
)
```

```
Commands completed successfully.

Completion time: 2020-05-23T12:33:36.5829701+05:30
```

TRUNCATE command in ddl



SQLQuery1.sql - SKANDIUM.pubs (SKANDIUM\Skandan KA (81))* - Microsoft SQL Server Management Studio

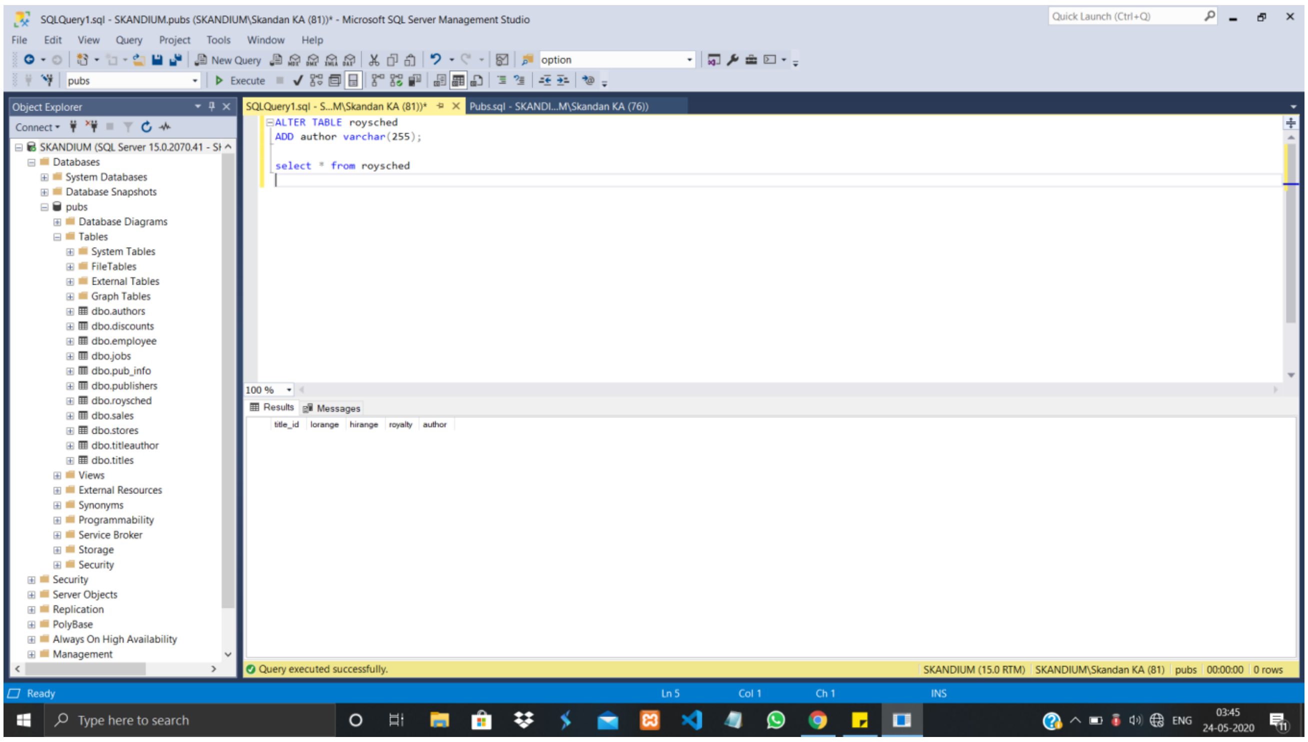
```
create table managers(
id int,
dept varchar(255)
);

truncate table managers;
```

Messages

```
Commands completed successfully.

Completion time: 2020-05-24T03:52:33.4882463+05:30
```

Query executed successfully. SKANDIUM (15.0 RTM) | SKANDIUM\Skandan KA (81) | pubs | 00:00:00 | 0 rows

ALTER command :



SQLQuery1.sql - SKANDIUM.pubs (SKANDIUM\Skandan KA (81))* - Microsoft SQL Server Management Studio

```
ALTER TABLE roysched
ADD author varchar(255);

select * from roysched
```

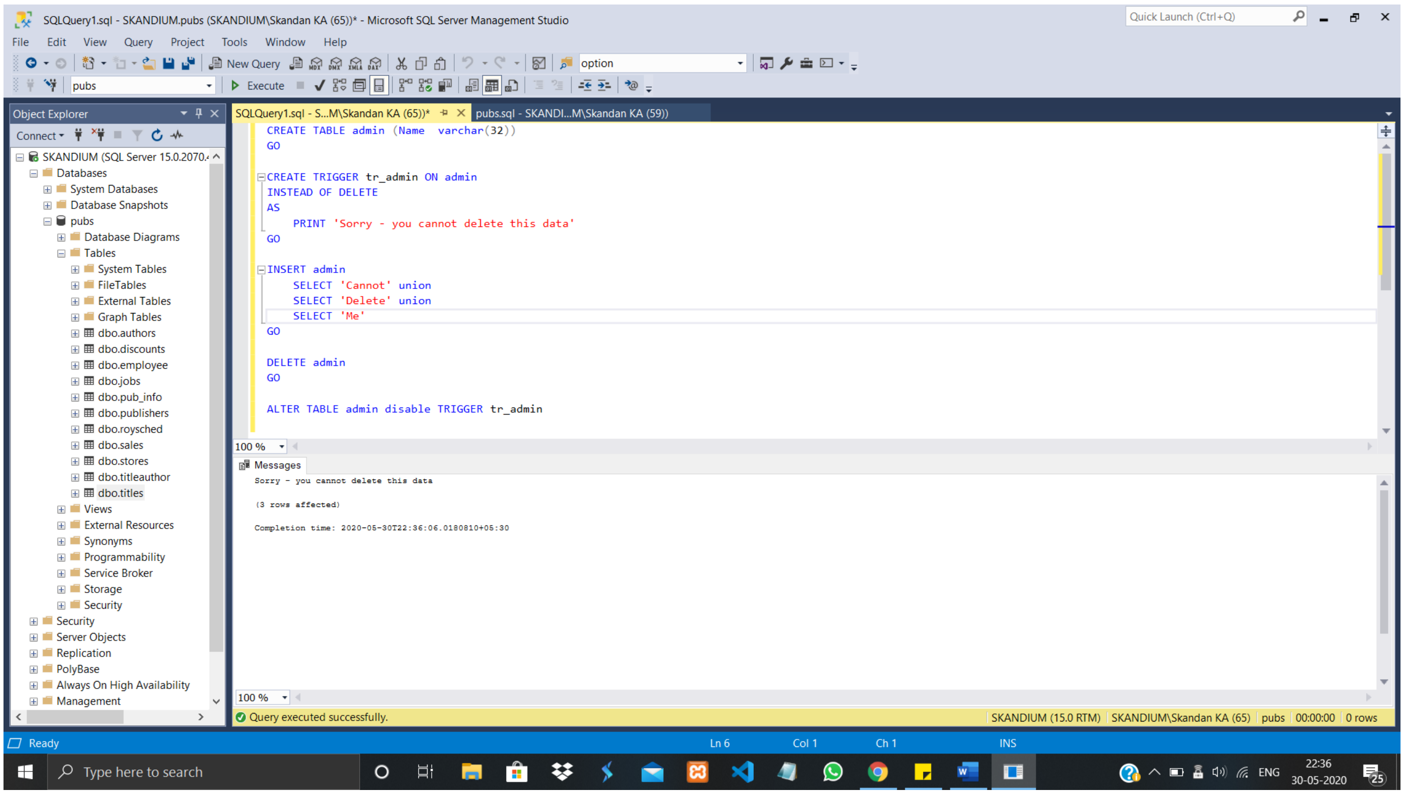
Results

title_id	lorange	hirange	royalty	author

Query executed successfully. SKANDIUM (15.0 RTM) | SKANDIUM\Skandan KA (81) | pubs | 00:00:00 | 0 rows

Triggers

```
CREATE TABLE admin (Name  varchar(32))
GO

CREATE TRIGGER tr_admin ON admin
INSTEAD OF DELETE
AS
    PRINT 'Sorry - you cannot delete this data'
GO

INSERT admin
    SELECT 'Cannot' union
    SELECT 'Delete' union
    SELECT 'Me'
GO

DELETE admin
GO
```



```
CREATE TABLE admin (Name  varchar(32))
GO

CREATE TRIGGER tr_admin ON admin
INSTEAD OF DELETE
AS
    PRINT 'Sorry - you cannot delete this data'
GO

INSERT admin
    SELECT 'Cannot' union
    SELECT 'Delete' union
    SELECT 'Me'
GO

DELETE admin
GO

ALTER TABLE admin disable TRIGGER tr_admin
```

```
Sorry - you cannot delete this data

(3 rows affected)

Completion time: 2020-05-30T22:36:06.0180810+05:30
```

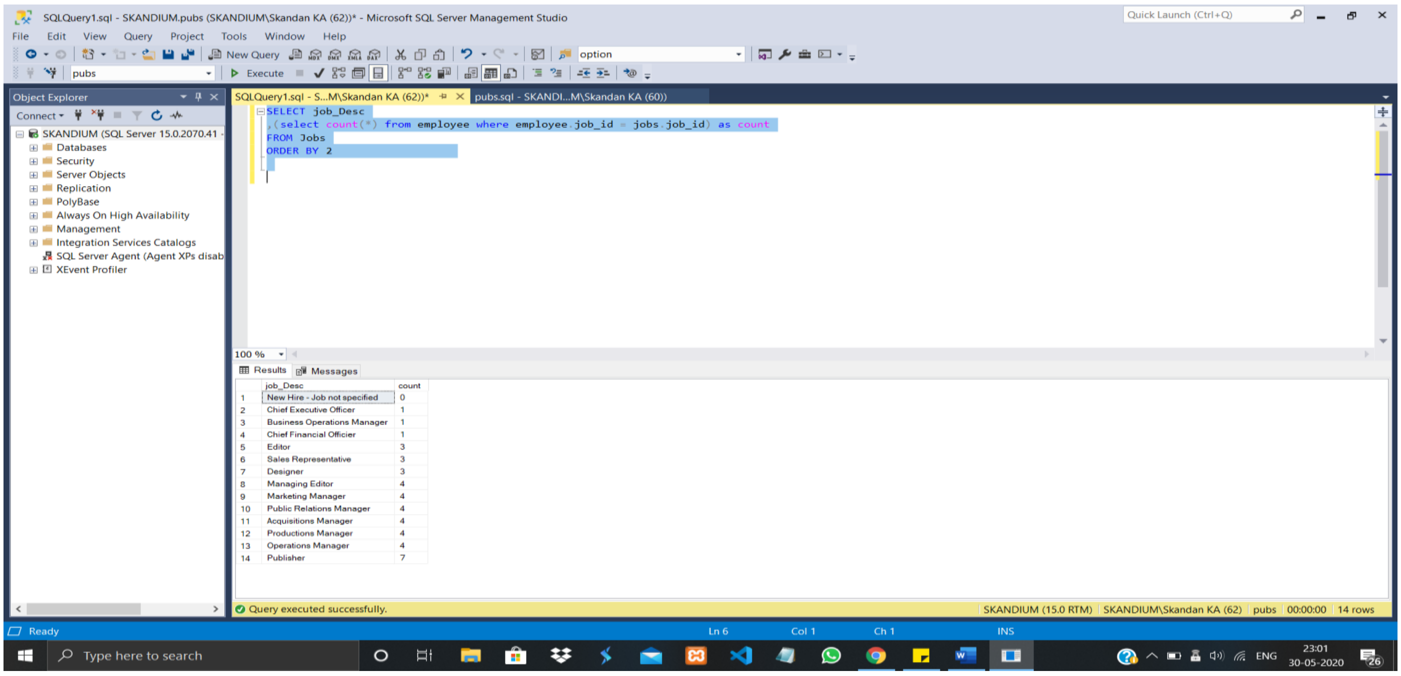
Query executed successfully.

Here a trigger tr_admin is created of type ‘instead’. If gives a printed message instead of deleting the table. The execution is showed in the above screenshot.

SQL Queries

Correlated Subquery to retrieve number of employees designated to a particular post.

```
SELECT job_Desc
,(select count(*) from employee where employee.job_id = jobs.job_id) as count
FROM Jobs
ORDER BY 2
```



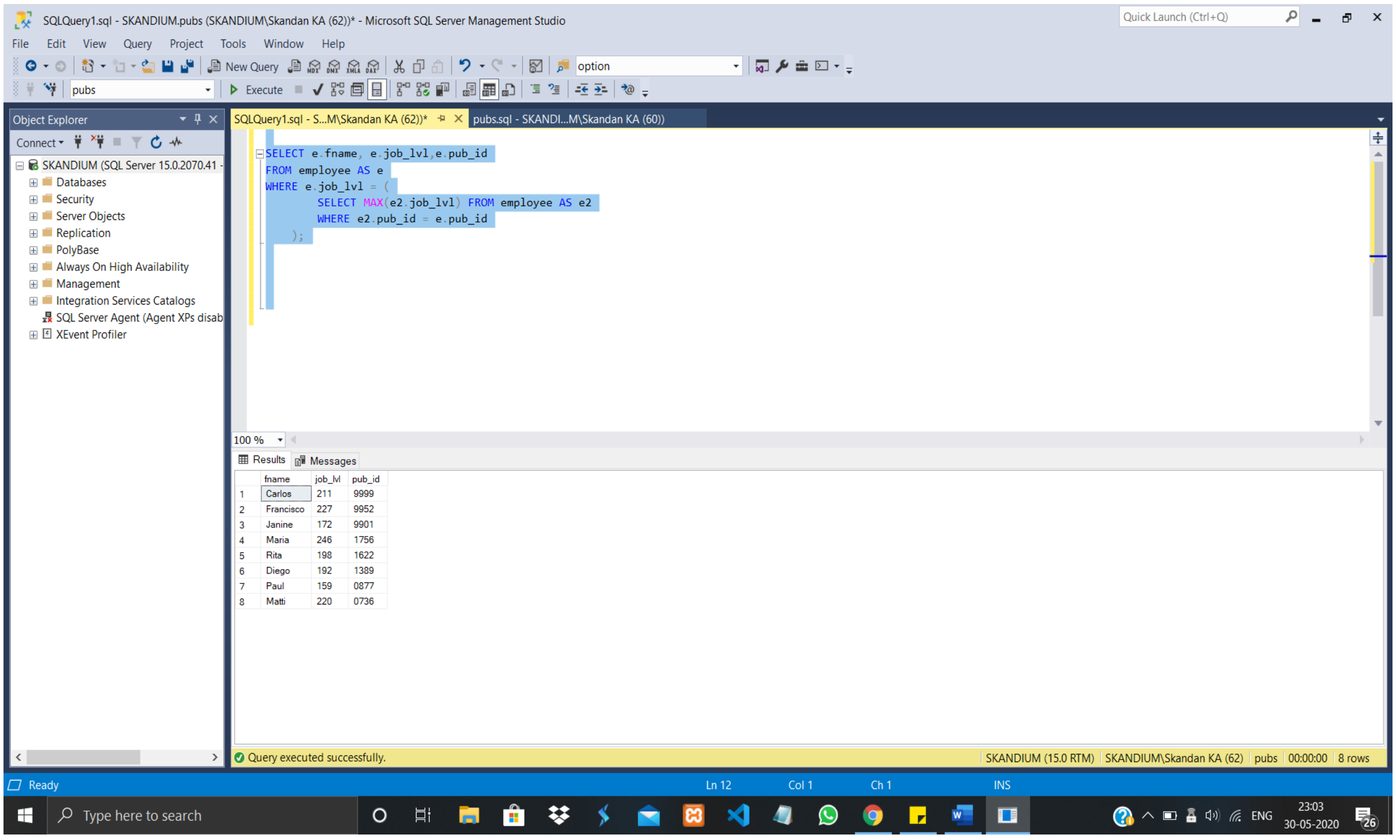
```
SELECT job_Desc
,(select count(*) from employee where employee.job_id = jobs.job_id) as count
FROM Jobs
ORDER BY 2
```

	job_Desc	count
1	New Hire - Job not specified	0
2	Chief Executive Officer	1
3	Business Operations Manager	1
4	Chief Financial Officier	1
5	Editor	3
6	Sales Representative	3
7	Designer	3
8	Managing Editor	4
9	Marketing Manager	4
10	Public Relations Manager	4
11	Acquisitions Manager	4
12	Productions Manager	4
13	Operations Manager	4
14	Publisher	7

Query executed successfully. SKANDIUM (15.0 RTM) SKANDIUM\Skandan KA (62) pubs 00:00:00 14 rows

Correlated SubQuery to retrieve employees with max job_lvl of same pub_id

```
SELECT e.fname, e.job_lvl,e.pub_id
FROM employee AS e
WHERE e.job_lvl = (
        SELECT MAX(e2.job_lvl) FROM employee AS e2
        WHERE e2.pub_id = e.pub_id
    );

       select * from employee
```



```
SELECT e.fname, e.job_lvl,e.pub_id
FROM employee AS e
WHERE e.job_lvl = (
        SELECT MAX(e2.job_lvl) FROM employee AS e2
        WHERE e2.pub_id = e.pub_id
    );
```

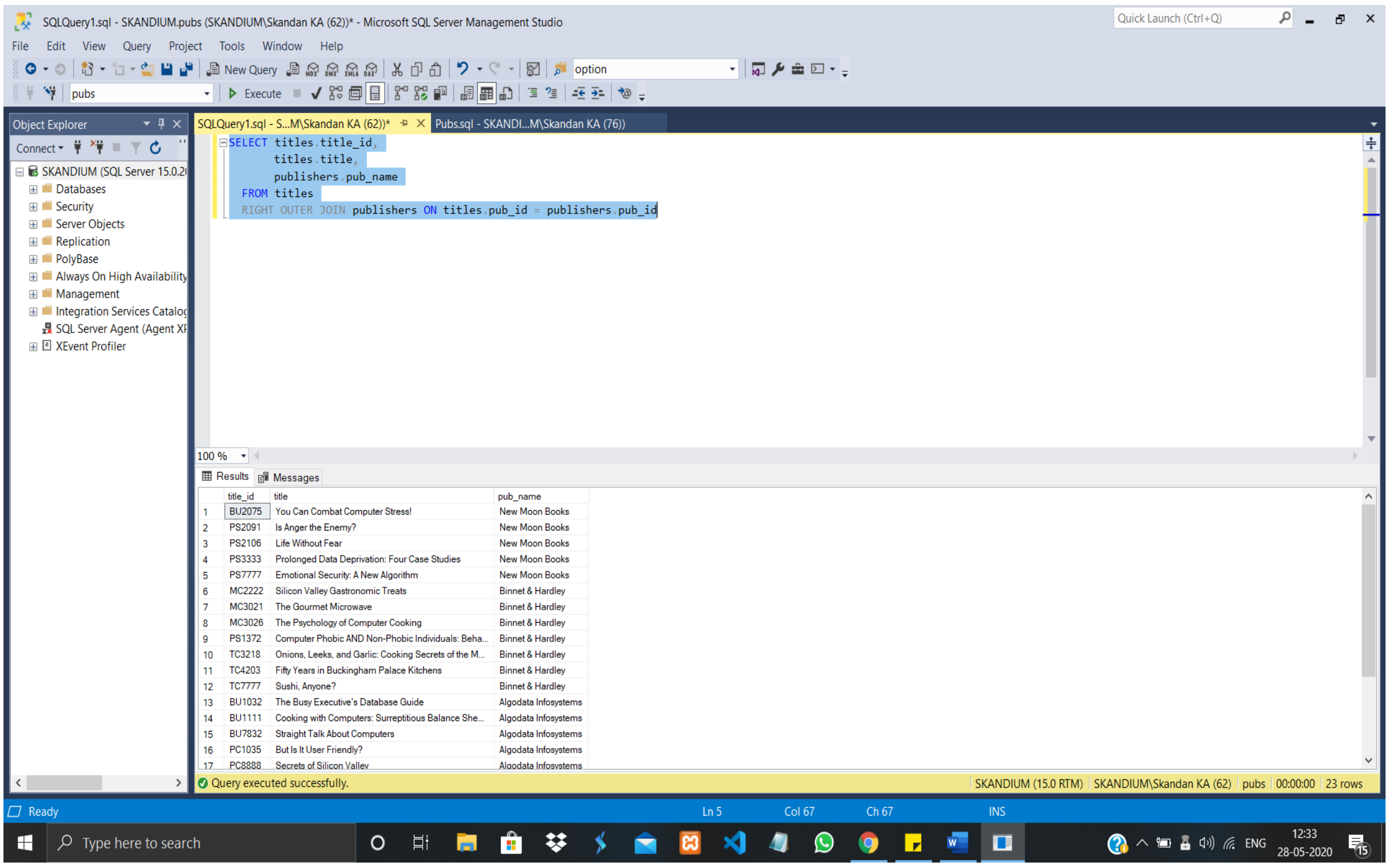
	fname	job_lvl	pub_id
1	Carlos	211	9999
2	Francisco	227	9952
3	Janine	172	9901
4	Maria	246	1756
5	Rita	198	1622
6	Diego	192	1389
7	Paul	159	0877
8	Matti	220	0736

Query executed successfully. SKANDIUM (15.0 RTM) SKANDIUM\Skandan KA (62) pubs 00:00:00 8 rows

JOINS:

RIGHT OUTER JOIN

```
SELECT titles.title_id,
       titles.title,
       publishers.pub_name
  FROM titles
  RIGHT OUTER JOIN publishers ON titles.pub_id = publishers.pub_id
```



```
SELECT titles.title_id,
      titles.title,
      publishers.pub_name
  FROM titles
  RIGHT OUTER JOIN publishers ON titles.pub_id = publishers.pub_id
```

	title_id	title	pub_name
1	BU2075	You Can Combat Computer Stress!	New Moon Books
2	PS2091	Is Anger the Enemy?	New Moon Books
3	PS2106	Life Without Fear	New Moon Books
4	PS3333	Prolonged Data Deprivation: Four Case Studies	New Moon Books
5	PS7777	Emotional Security: A New Algorithm	New Moon Books
6	MC2222	Silicon Valley Gastronomic Treats	Binnet & Hardley
7	MC3021	The Gourmet Microwave	Binnet & Hardley
8	MC3026	The Psychology of Computer Cooking	Binnet & Hardley
9	PS1372	Computer Phobic AND Non-Phobic Individuals: Beha...	Binnet & Hardley
10	TC3218	Onions, Leeks, and Garlic: Cooking Secrets of the M...	Binnet & Hardley
11	TC4203	Fifty Years in Buckingham Palace Kitchens	Binnet & Hardley
12	TC7777	Sushi, Anyone?	Binnet & Hardley
13	BU1032	The Busy Executive's Database Guide	Algodata Infosystems
14	BU1111	Cooking with Computers: Surreptitious Balance She...	Algodata Infosystems
15	BU7832	Straight Talk About Computers	Algodata Infosystems
16	PC1035	But Is It User Friendly?	Algodata Infosystems
17	PC8888	Secrets of Silicon Valley	Algodata Infosystems

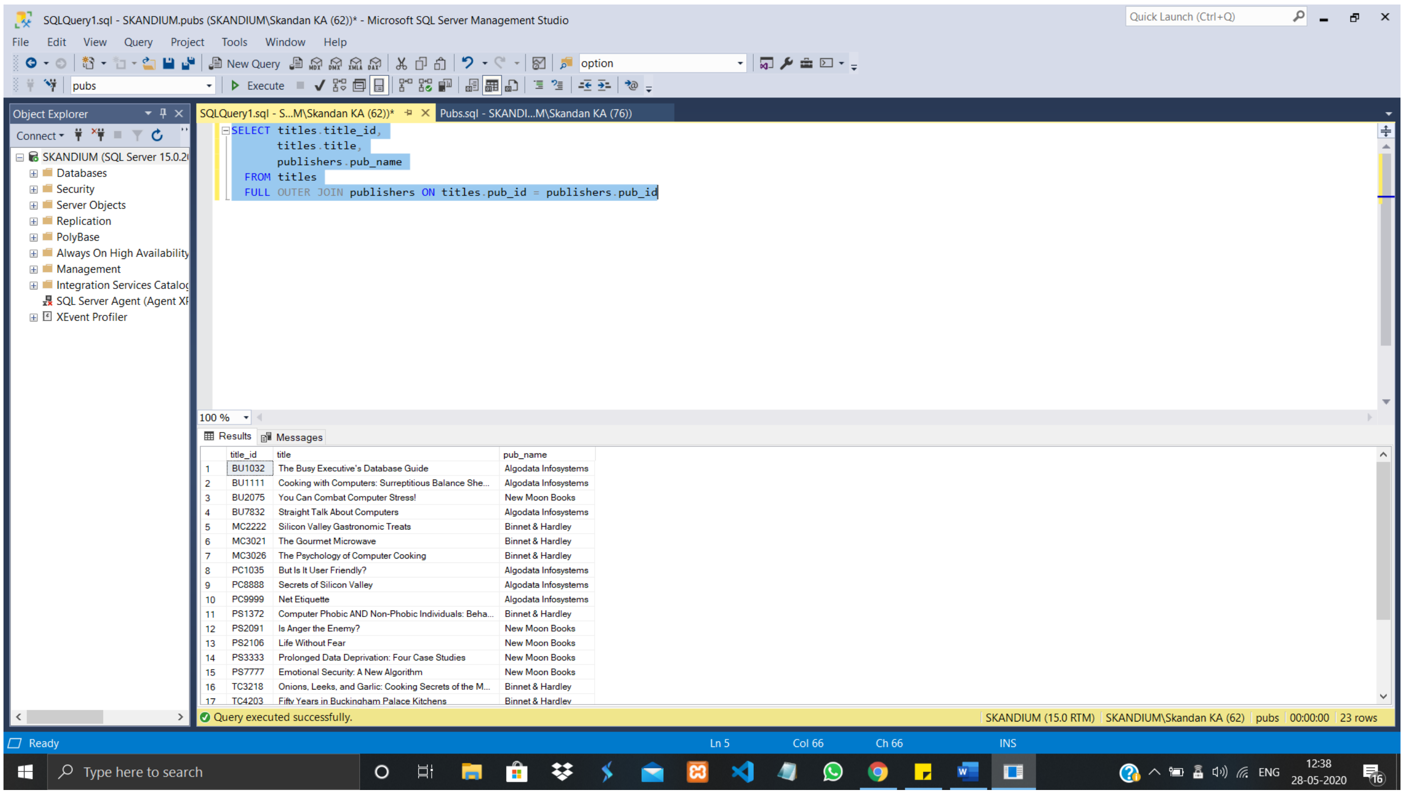
Query executed successfully.

SKANDIUM (15.0 RTM) | SKANDIUM\Skandan KA (62) | pubs | 00:00:00 | 23 rows

SQLQuery1.sql - SKANDIUM.pubs (SKANDIUM\Skandan KA (62))* - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
New Query
Execute
pubs
option
Object Explorer
SKANDIUM (SQL Server 15.0.2
Databases
Security
Server Objects
Replication
PolyBase
Always On High Availability
Management
Integration Services Catalog
SQL Server Agent (Agent XP
XEvent Profiler
SQLQuery1.sql - S...M\Skandan KA (62))*
Pubs.sql - SKANDI...M\Skandan KA (76))
100 %
Results
Messages
Ready
Ln 5
Col 67
Ch 67
INS
Type here to search
ENG
12:33
28-05-2020

FULL OUTER JOIN

```
SELECT titles.title_id,
       titles.title,
       publishers.pub_name
  FROM titles
  FULL OUTER JOIN publishers ON titles.pub_id = publishers.pub_id
```



```sql
SELECT titles.title_id,
       titles.title,
       publishers.pub_name
  FROM titles
  FULL OUTER JOIN publishers ON titles.pub_id = publishers.pub_id
```

	title_id	title	pub_name
1	BU1032	The Busy Executive's Database Guide	Algodata Infosystems
2	BU1111	Cooking with Computers: Surreptitious Balance She...	Algodata Infosystems
3	BU2075	You Can Combat Computer Stress!	New Moon Books
4	BU7832	Straight Talk About Computers	Algodata Infosystems
5	MC2222	Silicon Valley Gastronomic Treats	Binnet & Hardley
6	MC3021	The Gourmet Microwave	Binnet & Hardley
7	MC3026	The Psychology of Computer Cooking	Binnet & Hardley
8	PC1035	But Is It User Friendly?	Algodata Infosystems
9	PC8888	Secrets of Silicon Valley	Algodata Infosystems
10	PC9999	Net Etiquette	Algodata Infosystems
11	PS1372	Computer Phobic AND Non-Phobic Individuals: Beha...	Binnet & Hardley
12	PS2091	Is Anger the Enemy?	New Moon Books
13	PS2106	Life Without Fear	New Moon Books
14	PS3333	Prolonged Data Deprivation: Four Case Studies	New Moon Books
15	PS7777	Emotional Security: A New Algorithm	New Moon Books
16	TC3218	Onions, Leeks, and Garlic: Cooking Secrets of the M...	Binnet & Hardley
17	TC4203	Fifty Years in Buckingham Palace Kitchens	Binnet & Hardley

```
Query executed successfully.    SKANDIUM (15.0 RTM) | SKANDIUM\Skandan KA (62) | pubs | 00:00:00 | 23 rows
```

Conclusion

<
Write a few sentences about the capabilities of your system
Limitations and future enhancements
>

This Publishers database management system enhances the experience of journey of the books from publisher to the customer via stores. It can complete data regarding employees with their respective designations, number of copies received by stores, discounts etc.. so that all the mess could be avoided that would have arised if done manually.
Talking about some of the limitations, Complex data is difficult to locate and manage.
It is difficult to manage the relationships between various tables as many tables are present.
The administrator has to continuously monitor the system to make sure that the data is being updated correctly.
Further enhancements can be made like integrating a billing system to this so as to start making the sales in this system itself directly to the customer, thus making the system highly productive.
It could also be made to accommodate the reviews of the books read by the users which would be of great help to the authors for their future endevour.

Thank You.