目录

实验任务书（实验一）

实验内容：

1、 用 SQL 语句创建数据库 CAP，数据文件名为 CAPData.mdf，数据文件的初始存储空间大小为 50M，最大存储空间为 500M，存储空间自动增长量为 10M。

2、 在 CAP 数据库中用 SQL 语句创建下面的 4 张表，合理设计每个字段的数据类型，建立主键与外键约束。表 Products 中的 Price 字段不允许为空。表 Customers 的 discnt 字段取值范围在[0,30]之间。利用 SQL 语句向表中添加如下数据：

Customers

cid	cname	city	discnt
C001	TipTop	Duluth	10.00
C002	Basics	Dallas	12.00
C003	Allied	Dallas	8.00
C004	ACME	Duluth	8.00
C005	Oriental	Kyoto	6.00
C006	ACME	Kyoto	0.00

Products

Pid	pname	city	quantity	price
P01	comb	Dallas	111400	0.50
P02	brush	Newark	203000	0.50
P03	razor	Duluth	150600	1.00
P04	Pen	Duluth	125300	1.00
P05	pencil	Dallas	221400	1.00
P06	folder	Dallas	123100	2.00
P07	case	Newark	100500	1.00

Agents

Aid	aname	city	percent
A01	smith	New York	6
A02	Jones	Newark	6
A03	Brown	Tokyo	7
A04	Gray	New York	6
A05	Otasi	Duluth	5
A06	Smith	Dallas	5

Orders

Ordno	month	cid	aid	pid	qty	dollars

1011	Jan	C001	A01	P01	1000	450.00
1012	Jan	C001	A01	P01	1000	450.00
1019	Feb	C001	A02	P02	400	180.00
1017	Feb	C001	A06	P03	600	540.00
1018	Feb	C001	A03	P04	600	540.00
1023	Mar	C001	A04	P05	500	450.00
1022	Mar	C001	A05	P06	400	720.00
1025	Apr	C001	A05	P07	800	720.00
1013	Jan	C002	A03	P03	1000	880.00
1026	May	C002	A05	P03	800	704.00
1015	Jan	C003	A03	P05	1200	1104.00
1014	Jan	C003	A03	P05	1200	1104.00
1021	Feb	C004	A06	P01	1000	460.00
1016	Jan	C004	A01	P01	1000	500.00
1020	Feb	C005	A03	P07	600	600.00
1024	Mar	C006	A06	P01	800	400.00

3、 利用系统预定义的存储过程 sp_helpdb 查看数据库的相关信息，例如所有者、大小、创建日期等。

```
EXEC sp_helpdb CAP
GO
```

结果 消息

	name	db_size	owner	dbid	created	status	compatibility_level
1	CAP	122.00 MB	LAPTOP-IS2KHG9B\equ	5	05 9 2023	Status=ONLINE, Updateability=READ_WRITE, UserAc...	160

	name	fileid	filename	filegroup	size	maxsize	growth	usage
1	cap	1	D:\Sql\CAPData.mdf	PRIMARY	51200 KB	512000 KB	10240 KB	data only
2	CAP_log	2	D:\Sql\CAP_log.ldf	NULL	73728 KB	2147483648 KB	65536 KB	log only

4、 利用系统预定义的存储过程 sp_helpconstraint 查看表中出现的约束（包括 Primary key, Foreign key, check constraint, default, unique）

```
EXEC sp_helpconstraint Customers
GO
EXEC sp_helpconstraint Orders
GO
```

结果 消息

	Object Name
1	Customers

	constraint_type	constraint_name	delete_action	update_action	status_enabled	status_for_replication	constraint_keys
1	CHECK on column discnt	CK__customers__discn__398D8EEE	(n/a)	(n/a)	Enabled	Is_For_Replication	([discnt]>=(0) AND [discnt]<=(30))
2	PRIMARY KEY (clustered)	PK__customer__D837D05F35964C49	(n/a)	(n/a)	(n/a)	(n/a)	cid

	Table is referenced by foreign key
1	CAP.dbo.Orders: FK__Orders__cid__49C3F6B7

结果 消息

	Object Name
1	Orders

	constraint_type	constraint_name	delete_action	update_action	status_enabled	status_for_replication	constraint_keys
1	FOREIGN KEY	FK__Orders__aid__4AB81AF0	No Action	No Action	Enabled	Is_For_Replication	aid
2							REFERENCES CAP.dbo.Agents (Aid)
3	FOREIGN KEY	FK__Orders__cid__49C3F6B7	No Action	No Action	Enabled	Is_For_Replication	cid
4							REFERENCES CAP.dbo.Customers (cid)
5	FOREIGN KEY	FK__Orders__pid__4BAC3F29	No Action	No Action	Enabled	Is_For_Replication	pid
6							REFERENCES CAP.dbo.Products (Pid)
7	PRIMARY KEY ...	PK__Orders__67ACA39157...	(n/a)	(n/a)	(n/a)	(n/a)	Ordno

5、 创建一张表 Orders_Jan，表的结构与 Orders 相同，将 Orders 表中 month 为‘Jan’的订单记录复制到表 Orders_Jan 中。

```
select * into Orders_Jan
from Orders
where 1 = 0;
GO
```

结果 消息

	Ordno	month	cid	aid	pid	qty	dollars
1	1011	Jan	C001	A01	P01	1000	450
2	1012	Jan	C001	A01	P01	1000	450
3	1013	Jan	C002	A03	P03	1000	880
4	1014	Jan	C003	A03	P05	1200	1104
5	1015	Jan	C003	A03	P05	1200	1104
6	1016	Jan	C004	A01	P01	1000	500

6、 将 Orders 表中 month 为‘Jan’的订单记录全部删掉。

```
select * into Orders_Jan
from Orders
where 1 = 0;
GO
--始终返回 false,不会复制数据，只会复制表的结构--

insert into Orders_Jan
select * from Orders
where month = 'Jan'
GO

--检查
select * from Orders_Jan

delete from Orders
where month = 'Jan'
GO
```

7、 对曾经下过金额（dollars）大于 500 的订单的客户，将其 discnt 值增加 2 个

百分点（+2）。

```
update Customers
set discnt += 2
where (cid in (select Customers.cid
                    from Customers, Orders
                    where Customers.cid = Orders.cid and Orders.dollars > 500))
Go
```

8、 写一段 TSQL 程序，向表 Orders 中增加 5000 条记录，要求订单尽可能均匀地分布在 12 个月中。

具体思路：

（1） 声明了一系列变量，包括计数器@counter、订单号@Ordno、月份@mon、客户 ID@cid、代理商 ID@aid、产品 ID@pid、数量@qty 和金额@dollar。

（2） 进入循环，循环条件是 @counter <= 5000，表示循环次数为 5000 次。

（3） 在循环内部进行如下操作：

- 生成一个介于 1 到 12 之间的随机整数@randnum，用来表示月份。
- 根据随机整数@randnum，将对应的月份赋值给@mon。
- 从 Customers 表中随机选择一个客户 ID@cid。
- 从 Agents 表中随机选择一个代理商 ID@aid。
- 从 Products 表中随机选择一个产品 ID@pid。
- 生成一个介于 1 到 100 之间的随机整数@qty，表示订单的数量。
- 生成一个介于 0 到 1000 之间的随机浮点数@dollars，并将其转换为 NUMERIC 类型的数据，保留两位小数，表示订单的金额。
- 将生成的订单数据插入到 Orders 表中。
- 计数器@counter 加一，用于控制循环次数。
- 订单号@Ordno 加一，用于生成不同的订单号。

（4） 结束循环。

```
DECLARE @counter INT = 1,
        @Ordno INT = 1028,
        @mon CHAR(3),
        @cid CHAR(4),
        @aid CHAR(3),
        @pid CHAR(3),
        @qty INT,
        @dollars NUMERIC(10,2),
        @randnum INT;

WHILE @counter <= 5000
BEGIN
    SET @randnum = ABS(CHECKSUM(NewId())) % 12 + 1; -- 生成月份

    SET @mon =
```

```
            CASE @randnum
                WHEN 1 THEN 'Jan'
                WHEN 2 THEN 'Feb'
                WHEN 3 THEN 'Mar'
                WHEN 4 THEN 'Apr'
                WHEN 5 THEN 'May'
                WHEN 6 THEN 'Jun'
                WHEN 7 THEN 'Jul'
                WHEN 8 THEN 'Aug'
                WHEN 9 THEN 'Sep'
                WHEN 10 THEN 'Oct'
                WHEN 11 THEN 'Nov'
                WHEN 12 THEN 'Dec'
            END;

        SELECT TOP 1 @cid = cid FROM Customers ORDER BY NEWID(); -- 获取随机客户的
cid

        SELECT TOP 1 @aid = aid FROM Agents ORDER BY NEWID(); -- 获取随机代理商的
aid

        SELECT TOP 1 @pid = pid FROM Products ORDER BY NEWID(); -- 获取随机产品的
pid

        --生成一个介于 1 到 100 之间的随机整数
        SET @qty = ABS(CHECKSUM(NewId())) % 100 + 1; -- 生成数量

        --生成一个介于 0 到 1000 之间的随机浮点数，并将其转换为 NUMERIC 类型的数据，
保留两位小数
        SET @dollars = CAST(RAND() * 1000 AS NUMERIC(10,2)); -- 生成金额

        INSERT INTO Orders VALUES (CAST(@Ordno AS CHAR(4)), @mon, @cid, @aid, @pid,
@qty, @dollars); -- 插入新订单

        SET @counter += 1;
        -- 计数器加一
        SET @Ordno += 1;
END;
```

Ordno	month	cid	aid	pid	qty	dollars
1028	Feb	C349	A11	P31	68	30
1029	Oct	C298	A39	P72	4	271.81
1030	Jun	C892	A46	P28	31	940.91
1031	Mar	C640	A00	P89	6	577.3
1032	Dec	C474	A54	P15	28	370.92
1033	Feb	C657	A20	P92	11	675.81
1034	Aug	C642	A41	P17	9	65.35
1035	Oct	C602	A57	P45	30	993.95
1036	Aug	C903	A74	P88	13	499.13
1037	Oct	C372	A22	P27	83	991.68
1038	Apr	C359	A15	P41	71	528.15
1039	Jun	C947	A39	P03	14	371.65
1040	Aug	C212	A80	P61	75	986.04
1041	Aug	C345	A68	P99	33	649.33
1042	Jul	C447	A52	P18	92	784.8
1043	Dec	C734	A19	P09	93	142.05
1044	Sep	C252	A22	P41	86	746.47
1045	May	C726	A27	P76	17	159.5
1046	Jul	C579	A96	P44	4	98.36
1047	Jul	C628	A83	P77	100	538.38
1048	Oct	C474	A05	P74	22	591.5
1049	May	C008	A15	P93	90	396.71
1050	Apr	C110	A84	P01	8	16.83
1051	May	C726	A41	P42	26	252.13
1052	Sep	C340	A70	P06	43	482.59
1053	Jan	C354	A82	P61	63	874.09
1054	Jan	C228	A19	P36	35	728.51
1055	Jul	C672	A50	P19	79	299.77
1056	Apr	C550	A99	P56	54	750.76

9、 在表 Orders 的’month’字段上建立索引。

```
create index month_idx on Orders(month)
```

10、创建一个视图 order_month_summary，视图中的字段包括月份、该月的订单总量和该月的订单总金额。基于视图 order_month_summary，查询第一季度各个月份的订单总量和订单总金额。

```
create view order_month_summary as
select Orders.month as month,COUNT(*) as total_orders,SUM(Orders.dollars) as sum_dollars
from Orders
group by Orders.month

--查询--
select month,total_orders,sum_dollars
from order_month_summary
```

结果 消息

	month	total_orders	sum_dollars
1	Jun	372	191547.539746642
2	Jul	430	218927.17052865
3	Oct	424	212734.600066662
4	Sep	450	228249.640399992
5	Dec	443	221441.219964504
6	Feb	424	212441.859755635
7	Mar	403	197864.57959944
8	Apr	425	204837.480385482
9	May	402	198408.589898109
10	Aug	401	199714.889834642
11	Jan	444	219447.350283265
12	Nov	382	193218.26025486

实验任务书（实验二）

实验内容：

一、熟悉示例数据库

1．运行给定的 SQL Script，建立数据库 GlobalToyz。

2．创建数据库关系图，了解表的结构。

3．在 Orders 表中增加 1000 笔订单数据，注意与其它表之间的关系。

具体思路：

（1） 声明了一系列变量，包括订单号、玩具 ID、订单日期等等。

（2） 进入循环，循环条件是 @i<=1011，每次循环 @i 的值增加 1，直到达到循环结束条件。

（3） 在循环内部进行如下操作：

- 生成 6 位长度的订单号，格式为以 0 开头的数字。
- 根据当前循环次数 @i 计算出订单产生日期。
- 从 ShoppingCart 表中随机选择一个购物车 ID。
- 从 Toys 表中随机选择一个玩具 ID 和玩具单价。
- 从 Shopper 表中随机选择一个购物者 ID。
- 从 ShippingMode 表中随机选择一个玩具运输种类。
- 根据选择的玩具 ID 和购物车 ID，在 ShoppingCart 表中找到对应的玩具重量。
- 根据购物者 ID 在 Shopper 表中找到对应的购物者家乡号 ID。
- 根据玩具运输种类和购物者家乡号 ID，在 ShippingRate 表中找到每磅运输的单价。
- 根据每磅运输的单价和玩具质量计算出运输费用。
- 在 Wrapper 表中随机选择一个包装单价和包装种类 ID。
- 设置订单是否被处理的标识为 'Y'。
- 随机生成玩具是否需要包装费用的标识。
- 根据玩具是否需要包装费用，设置玩具的包装种类 ID 和包装费用。
- 计算玩具的总价值，即玩具单价乘以玩具质量。
- 计算玩具的总费用，包括玩具价值、运输费用和包装费用。
- 根据玩具运输种类在 ShippingMode 表中找到最大延迟时间。
- 根据最大延迟时间和订单产生日期计算预计交付日期。
- 将生成的订单数据插入到 Orders 表中。
- 将生成的订单详情数据插入到 OrderDetail 表中。

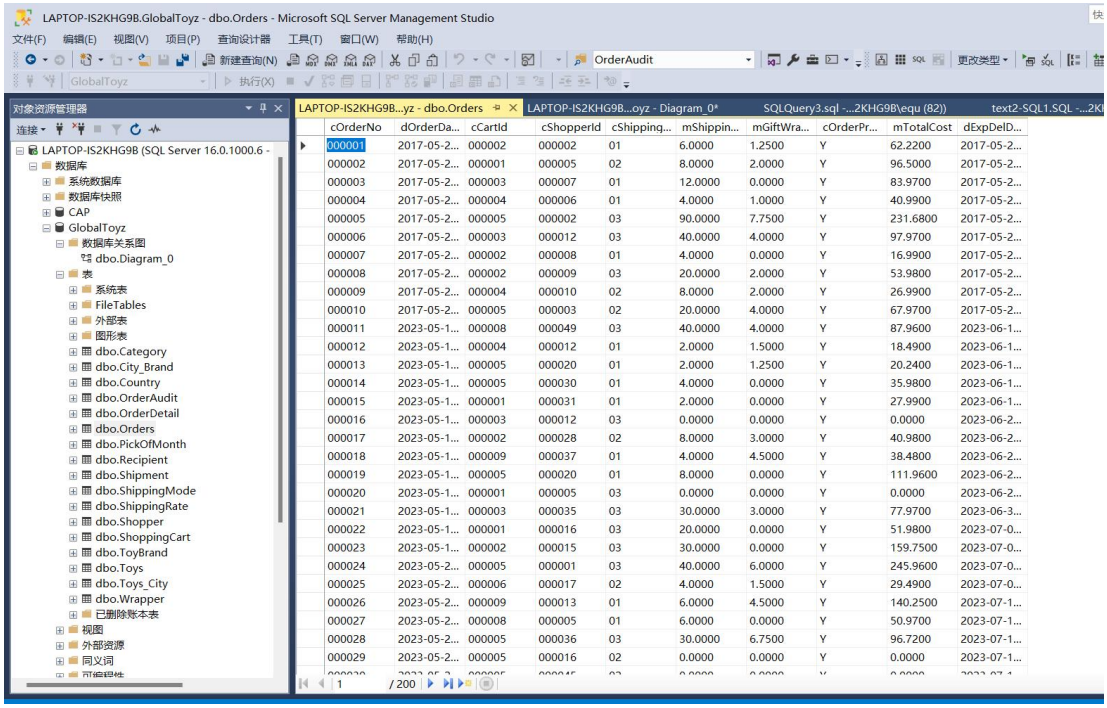
（4） 结束循环。

```
DECLARE @i int = 11, @cOrderNo as char(6), @cToyId as char(6), @dOrderDate as datetime,
@cShopperId as char(6), @cCartId as char(6),
@cShippingModeId as char(2), @mShippingCharges as money, @mGiftWrapCharges as money,
@cOrderProcessed as char(1), @mTotalCost as money, @dExpDelDate as datetime, @siQty as
smallint, @cWrapperId as char(3), @vMessage as varchar(256), @toycost as money,
@minTotalCost as money, @sum as money, @shippingRate as money,
@coutryID as char(3), @wrapperRate as money, @GiftWrapper as char(1), @cToyRate as money,
```

```
@iMaxDelay int;
WHILE (@i<=1011)
BEGIN
    --订单号
    --RIGHT:截取该字符串的右侧部分（也就是末尾），并在其左侧添加 0，直到总长度为
6 位
    set @cOrderNo = (RIGHT('000000' + CAST(@i AS VARCHAR(6)), 6))

    --订单产生日期
    set @dOrderDate = DATEADD(hour, @i, '2023-05-19 00:00:00')

    --从表中选择一个购物车 id
    select @cCartId = cCartId
    from ShoppingCart
    order by newid()

    --从表中选择一个玩具 id，玩具单价
    select @cToyId = cToyId, @cToyRate = mToyRate
    from Toys
    order by newid()

    --从表中选择一个购物者 id
    select @cShopperId = cShopperId
    from Shopper
    order by newid()

    --从表中选择一个玩具运输种类
    select @cShippingModeId = cModeId
    from ShippingMode
    order by newid()

    --从表中找到玩具质量
    select @siQty = ShoppingCart.siQty
    from ShoppingCart
    where ShoppingCart.cToyId = @cToyId and ShoppingCart.cCartId = @cCartId
    --set @siQty = CAST(rand() * 5 as int)

    --购物者家乡号 Id 由购物者 Id 确定
    select @coutryID = Shopper.cCountryId
    from Shopper
    where Shopper.cShopperId = @cShopperId

    --每个玩具的每磅运输的单价
    select @shippingRate = mRatePerPound
```

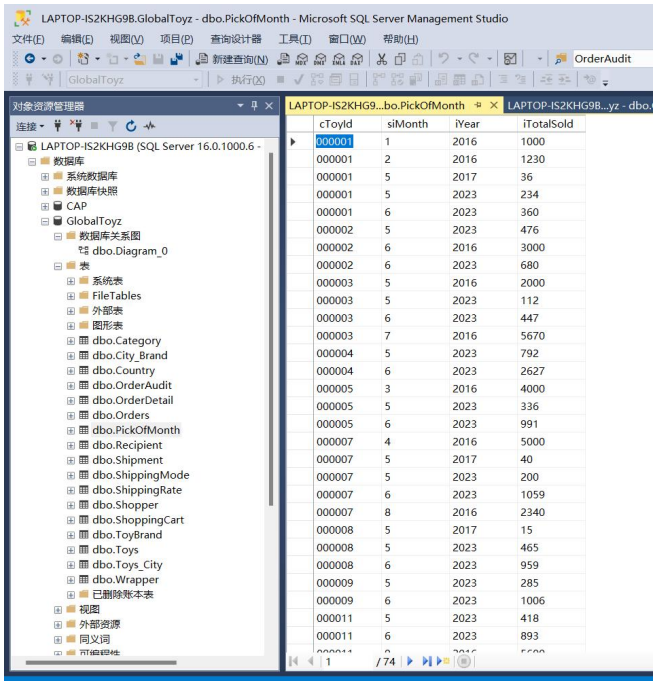
```
    from ShippingRate
    where ShippingRate.cModeId = @cShippingModeId and ShippingRate.cCountryID =
@coutryID

    --购物总价=每磅运输的单价*玩具质量
    set @mShippingCharges = @shippingRate * @siQty

    --处理 orderDetail 表,先随机取出包装单价和包装种类 Id
    select @wrapperRate = mWrapperRate, @cWrapperId = cWrapperId
    from Wrapper
    order by newid()

    --订单是否被处理：Y-已处理
    set @cOrderProcessed = 'Y'

    --标识玩具是否需要包装费用   Y-需要     N-不需要
    set @GiftWrapper = (SELECT CASE WHEN RAND() < 0.5 THEN 'Y' ELSE 'N' END)

    --玩具的包装种类 Id
    set @cWrapperId =
        case
            when @GiftWrapper = 'Y' then @cWrapperId
            when @GiftWrapper = 'N' then NULL
        end

    --玩具的包装费=玩具单个包装费用*数量
    set @mGiftWrapCharges =
        case
            when @GiftWrapper = 'Y' then @wrapperRate * @siQty
            when @GiftWrapper = 'N' then 0
        end

    --玩具价值=玩具单价*质量
    --由于这里是一对一，只需要产生一条
    set @ToyCost = @cToyRate * @siQty

    --玩具总费用=玩具价值+运输费用+包装费用
    set @mTotalCost = @ToyCost+ @mShippingCharges + @mGiftWrapCharges

    --交付时间：最大延迟时间由 ShippingMode 寻找
    select @iMaxDelay
    from ShippingMode
    where @cShippingModeId = ShippingMode.iMaxDelDays
    set @dExpDelDate =    DATEADD(day, @iMaxDelay, '2023-05-19 00:00:00') --预计交付日
```

```
期

    --插入数据到 Orders 表
    INSERT INTO Orders
    VALUES (@cOrderNo, @dOrderDate, @cCartId, @cShopperId, @cShippingModeId,
@mShippingCharges,
        @mGiftWrapCharges, @cOrderProcessed, @mTotalCost, @dExpDelDate)

    set @vMessage = NULL

    --插入数据到 OrderDetail 表
    INSERT INTO OrderDetail (cOrderNo, cToyId, siQty, cGiftWrap, cWrapperId, vMessage,
mToyCost)
    VALUES (
        @cOrderNo, -- cOrderNo
        @cToyId, -- cToyId
        @siQty, -- siQty
        @GiftWrapper, -- cGiftWrap
        @cWrapperId, -- cWrapperId
        @vMessage, -- vMessage
        @toycost
    );

    SET @i = @i + 1
END
GO
```



4．在 Orders 表中新增数据之后，更新 PickofMonth 表。

```
INSERT INTO PickOfMonth(cToyId, siMonth, iYear, iTotalSold)
SELECT cToyId, MONTH(dOrderDate), YEAR(dOrderDate),sum(mToyCost)
FROM OrderDetail
JOIN Orders ON OrderDetail.cOrderNo = Orders.cOrderNo
GROUP BY cToyId, MONTH(dOrderDate), YEAR(dOrderDate)
GO
```



二、查询、更新数据库

1. 查找属于 California 和 Florida 州的顾客的名、姓和 emailID。

```
select vFirstName, vLastName, vEmailId
from Shopper
where Shopper.cState ='California' or Shopper.cState='Florida'
```

	vFirstName	vLastName	vEmailId
1	Barbara	Johnson	barbaraj@speedmail.com
2	Catherine	Roberts	catheriner@qmail.com
3	Charles	Brown	charlesb@speedmail.com
4	Cynthia	Miller	cynthiam@qmailcom
5	David	Moore	davidm@qmail.com
6	Joseph	Martinez	josephm@qmail.com
7	Patricia	Wright	patreciaw@speedmail.com
8	Paul	Lopez	paull@qmail.com
9	Robert	Scott	Roberts@speedmail.com
10	Sandra	Adams	Sandra@qmail.com
11	Sarah	Baker	sarahb@qmail.com
12	David	Cooper	davidc@speedmail.com
13	John	Doran	johnd@qmail.com

2. 查找定单号码、顾客 ID，定单的总价值，并以定单的总价值的升序排列。

```
select cOrderNo,cShopperId, mTotalCost
from Orders
Order by Orders.mTotalCost
```

	cOrderNo	cShopperId	mTotalCost
184	000988	000011	0.00
185	000989	000003	0.00
186	000990	000042	0.00
187	000985	000034	0.00
188	000986	000005	0.00
189	000994	000020	0.00
190	000998	000041	0.00
191	001004	000017	0.00
192	001005	000046	0.00
193	001006	000006	0.00
194	001007	000008	0.00
195	001002	000019	0.00
196	000490	000026	8.99
197	000740	000030	9.99
198	000888	000015	10.49
199	000850	000014	10.99
200	000704	000005	10.99
201	000632	000032	10.99
202	000376	000007	10.99
203	000329	000034	10.99
204	000258	000003	10.99
205	000183	000025	10.99

	cOrderNo	cShopperId	mTotalCost
798	000511	000024	83.97
799	000401	000045	83.97
800	000003	000007	83.97
801	000778	000014	83.97
802	000816	000043	84.96
803	000341	000039	84.96
804	000551	000028	85.96
805	000763	000040	85.96
806	000719	000001	85.96
807	000875	000034	85.96
808	000847	000035	86.97
809	000680	000006	86.97
810	000668	000012	86.97
811	000622	000011	86.97
812	000055	000050	86.97
813	000285	000022	86.97
814	000237	000050	86.97
815	000129	000043	87.72
816	000455	000036	87.72
817	000122	000008	87.96
818	000087	000018	87.96
819	000011	000049	87.96
820	000619	000013	87.96
821	000858	000022	87.96
822	000796	000028	87.96
823	000456	000017	88.47
824	000917	000038	89.96

3. 查找在 orderDetail 表中 vMessage 为空值的行。

```
select *
from OrderDetail
where OrderDetail.vMessage is NULL
```

	cOrderNo	cToyId	siQty	cGiftWrap	cWrapperId	vMessage	mToyCost
1	000001	000007	2	N	NULL	NULL	39.98
2	000003	000017	3	N	NULL	NULL	71.97
3	000007	000006	1	N	NULL	NULL	12.99
4	000011	000012	4	Y	001	NULL	43.96
5	000012	000032	1	Y	005	NULL	14.99
6	000013	000018	1	Y	002	NULL	16.99
7	000014	000023	2	N	NULL	NULL	31.98
8	000015	000024	1	N	NULL	NULL	25.99
9	000016	000016	0	Y	006	NULL	0.00
10	000017	000032	2	Y	005	NULL	29.98
11	000018	000032	2	Y	006	NULL	29.98
12	000019	000024	4	N	NULL	NULL	103.96
13	000020	000015	0	N	NULL	NULL	0.00
14	000021	000008	3	Y	007	NULL	44.97
15	000022	000005	2	N	NULL	NULL	31.98
16	000023	000016	3	N	NULL	NULL	129.75
17	000024	000015	4	Y	003	NULL	199.96
18	000025	000017	1	Y	005	NULL	23.99
19	000026	000016	3	Y	005	NULL	129.75
20	000027	000029	3	N	NULL	NULL	44.97
21	000028	000007	3	Y	006	NULL	59.97
22	000029	000029	0	N	NULL	NULL	0.00
23	000030	000001	0	Y	005	NULL	0.00

4. 查找玩具名字中有“Racer”字样的所有玩具的基本资料。

```
select *
from Toys
where Toys.vToyName like '%Racer%'
```

	cToyId	vToyName	vToyDescription	cCategoryId	mToyRate	cBrandId	imPhoto	siToyQoh	siLowerAge	siUpperAge	siToyWeight	vToyImgPath
1	000027	X-90 Racers Set	The fast-paced action racing track is the ultim...	005	19.99	001	NULL	77	5	9	1	NULL
2	000028	Dune Racer	A set of dune buggies with a racing track.	005	9.99	004	NULL	78	4	9	1	NULL

5. 根据 2016 年的玩具销售总数，查找“Pick of the Month”玩具的前五名玩具的 ID。

```
select top 5 cToyId
from PickOfMonth
where PickOfMonth.iYear = '2016'
group by PickOfMonth.cToyId
order by sum(PickOfMonth.iTotalSold) desc
```

	cToyId
1	000003
2	000007
3	000026
4	000011
5	000021

6. 根据 OrderDetail 表，查找玩具总价值大于￥50 的定单的号码和玩具总价值。

```
select cOrderNo, SUM(mToyCost) as ToyCost
from OrderDetail
group by cOrderNo
having SUM(mToyCost) > 50
```

	cOrderNo	ToyCost
1	000001	54.97
2	000002	86.50
3	000003	71.97
4	000005	133.93
5	000006	53.97
6	000019	103.96
7	000023	129.75
8	000024	199.96
9	000026	129.75
10	000028	59.97

7. 查找一份包含所有装运信息的报表，包括：Order Number, Shipment Date, Actual Delivery Date, Days in Transit. (提示：Days in Transit = Actual Delivery Date – Shipment Date)

```
select cOrderNo, dShipmentDate, dActualDeliveryDate, (dActualDeliveryDate-dShipmentDate)
as Days_in_Transit
from Shipment
```

	cOrderNo	dShipmentDate	dActualDeliveryDate	Days_in_Transit
1	000001	2017-05-23 00:00:00.000	2017-05-24 00:00:00.000	1900-01-02 00:00:00.000
2	000002	2017-05-23 00:00:00.000	2017-05-23 00:00:00.000	1900-01-01 00:00:00.000
3	000003	2017-05-23 00:00:00.000	NULL	NULL
4	000004	2017-05-24 00:00:00.000	2017-05-26 00:00:00.000	1900-01-03 00:00:00.000
5	000005	2017-05-24 00:00:00.000	2017-05-25 00:00:00.000	1900-01-02 00:00:00.000
6	000006	2017-05-22 00:00:00.000	2017-05-23 00:00:00.000	1900-01-02 00:00:00.000
7	000007	2017-05-25 00:00:00.000	NULL	NULL
8	000008	2017-05-24 00:00:00.000	2017-05-24 00:00:00.000	1900-01-01 00:00:00.000
9	000009	2017-05-24 00:00:00.000	2017-05-25 00:00:00.000	1900-01-02 00:00:00.000
10	000010	2017-05-26 00:00:00.000	2017-05-28 00:00:00.000	1900-01-03 00:00:00.000

8. 查找所有玩具的名称、商标和种类（Toy Name, Brand, Category）。

```
select vToyName,cBrandName,cCategory
from Toys, ToyBrand, Category
where Toys.cBrandId = ToyBrand.cBrandId and Toys.cCategoryId = Category.cCategoryId
```

	vToyName	cBrandName	cCategory
1	Robby the Whale	Bobby	Activity
2	Water Channel System	Bobby	Activity
3	Parachute and Rocket	The Bernie Kids	Activity
4	Super Deluge	LAMOBIL	Activity
5	Light Show Lamp	Bobby	Dolls
6	Glass Decoration	Largo	Dolls
7	Tie Dye Kit	Frances-Price	Dolls
8	Alice in Wonderland	Bobby	Dolls
9	Glamorous Doll	Bobby	Dolls
10	Sleeping Beauty Doll	LAMOBIL	Dolls
11	Pet Loving Doll	Bobby	Dolls
12	Beautifull Hair Doll	The Bernie Kids	Dolls
13	Flower Loving Doll	Largo	Dolls
14	Victorian Dollhouse	The Bernie Kids	Dolls
15	Kitchen Set	Frances-Price	Pretend Play
16	Childrens Bedroom	LAMOBIL	Pretend Play
17	Nursery	Bobby	Pretend Play
18	Victorian Family	Bobby	Pretend Play
19	Birthday Party	Largo	Pretend Play
20	Tin Drum	Bobby	Musical toys

9. 查找玩具的名称和所有玩具的购物车 ID。如果玩具不在购物车中，也需在结果中出现。

```
select vToyName, cCartId
from Toys left join ShoppingCart
on Toys.cToyId = ShoppingCart.cToyId
```

	vToyName	cCartId
1	Robby the Whale	000001
2	Robby the Whale	000005
3	Water Channel System	NULL
4	Parachute and Rocket	NULL
5	Super Deluge	000004
6	Light Show Lamp	NULL
7	Glass Decoration	000007
8	Tie Dye Kit	000001
9	Alice in Wonderland	000001
10	Glamorous Doll	000002
11	Sleeping Beauty Doll	NULL
12	Pet Loving Doll	NULL
13	Beautifull Hair Doll	000006
14	Flower Loving Doll	NULL
15	Victorian Dollhouse	000002
16	Kitchen Set	000003
17	Kitchen Set	000006
18	Childrens Bedroom	000005
19	Childrens Bedroom	000009
20	Nursery	NULL

10. 以下列格式查找所有购物者的名字和他们的简称：（Initials, vFirstName, vLastName）,例如 Angela Smith 的 Initials 为 A.S。

```
select SUBSTRING(vFirstName,1,1)+'.' +SUBSTRING(vLastName,1,1)as Initials, vFirstName,
vLastName
from Shopper
```

	Initials	vFirstName	vLastName
1	A.S	Angela	Smith
2	B.J	Barbara	Johnson
3	B.W	Betty	Williams
4	C.J	Carol	Jones
5	C.R	Catherine	Roberts
6	C.B	Charles	Brown
7	C.D	Christo...	Davis
8	C.M	Cynthia	Miller
9	D.W	Daniel	Wilson
10	D.M	David	Moore
11	D.T	Deborah	Taylor
12	D.A	Donna	Anderson
13	D.T	Dorothy	Thomas

11. 查找所有玩具的平均价格，并舍入到整数。

```
select cast(ROUND(AVG(mToyRate),0)as int)
```

```
from Toys
```

	(无列名)
1	20

12. 查找所有购买者和收货人的名、姓、地址和所在城市，要求保留结果中的重复记录。

```
select ALL s.vFirstName, s.vLastName, s.vAddress, s.cCity, r.vFirstName, r.vLastName,
r.vAddress, r.cCity
from Orders o, Shopper s, Recipient r
where o.cShopperId = s.cShopperId and o.cOrderNo = r.cOrderNo
```

	vFirstName	vLastName	vAddress	cCity	vFirstName	vLastName	vAddress	cCity
1	Barbara	Johnson	227 Beach Ave.	Sunnyvale	Barbara	Johnson	227 Beach Ave.	Sunnyvale
2	Catherine	Roberts	5508 Aquiline Court	San Jose	Catherine	Roberts	5508 Aquiline Court	San Jose
3	Christopher	Davis	4896 11th ST	Hill Avenue	Christopher	Davis	4896 11th ST	Hill Avenue
4	Charles	Brown	7822 S. Glitzy Avenue	Maitland	Jennifer	Martin	9812 76th Street	Brooklyn
5	Barbara	Johnson	227 Beach Ave.	Sunnyvale	Barbara	Johnson	227 Beach Ave.	Sunnyvale
6	Donna	Anderson	7930 Orange St.	Las Vegas	Donna	Anderson	7930 Orange St.	Las Vegas
7	Cynthia	Miller	98066 Weary Storm Street	Moon Park	Laura	Rodriguez	3242 Limestone	WayMarietta
8	Daniel	Wilson	4642 Peripheral Drive	Brecksville	Michelle	Hernandez	1353 Realm Lakes	Naperville
9	David	Moore	8808 Joviality Drive	San Ramon	David	Moore	8808 Joviality Drive	San Ramon
10	Betty	Williams	1 Tread Road	Virginia Beach	Betty	Williams	1 Tread Road	Virginia Beach

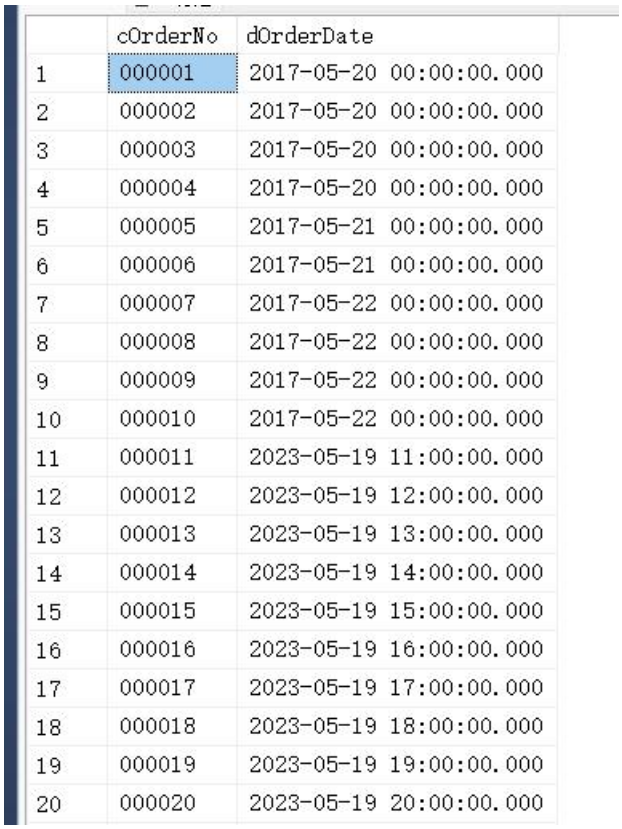
13. 查找没有包装的所有玩具的名称。（要求用子查询实现）

```
select vToyName
from Toys
where cToyId in (
select cToyId
from OrderDetail
where OrderDetail.cWrapperId is NULL)
```

	vToyName
1	Nursery
2	Parachute and Rocket
3	Dune Racer
4	Glass Decoration
5	Sleeping Beauty Doll
6	Glamorous Doll
7	My First Flashlight
8	Victorian Dollhouse
9	Robby the Whale
10	Large Duck
11	Super Deluge
12	Tin Drum
13	Baby Minnie

14. 查找已收货定单的定单号码以及下定单的时间。（要求用子查询实现）

```
select cOrderNo, dOrderDate
from Orders
where cOrderNo in(
      select cOrderNo
      from Orders
      where Orders.cOrderProcessed = 'Y'
)
```



	cOrderNo	dOrderDate
1	000001	2017-05-20 00:00:00.000
2	000002	2017-05-20 00:00:00.000
3	000003	2017-05-20 00:00:00.000
4	000004	2017-05-20 00:00:00.000
5	000005	2017-05-21 00:00:00.000
6	000006	2017-05-21 00:00:00.000
7	000007	2017-05-22 00:00:00.000
8	000008	2017-05-22 00:00:00.000
9	000009	2017-05-22 00:00:00.000
10	000010	2017-05-22 00:00:00.000
11	000011	2023-05-19 11:00:00.000
12	000012	2023-05-19 12:00:00.000
13	000013	2023-05-19 13:00:00.000
14	000014	2023-05-19 14:00:00.000
15	000015	2023-05-19 15:00:00.000
16	000016	2023-05-19 16:00:00.000
17	000017	2023-05-19 17:00:00.000
18	000018	2023-05-19 18:00:00.000
19	000019	2023-05-19 19:00:00.000
20	000020	2023-05-19 20:00:00.000

15. 查找一份基于 Orderdetail 的报表，包括 cOrderNo,cToyId 和 mToyCost，记录以 cOrderNo 升序排列，并计算每一笔定单的玩具总价值。

```
select cOrderNo, cToyId
from OrderDetail
order by cOrderNo

select cOrderNo, sum(mToyCost) as ToyCost
from OrderDetail
group by cOrderNo
order by cOrderNo
```

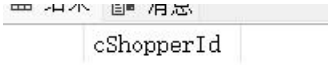
	cOrderNo	cToyId
1	000001	000007
2	000001	000008
3	000002	000016
4	000003	000017
5	000004	000030
6	000005	000001
7	000005	000024
8	000005	000030
9	000006	000013
10	000006	000017
11	000007	000006
12	000008	000023
13	000009	000018
14	000010	000020
15	000010	000021
16	000011	000012
17	000012	000032
18	000013	000018
19	000014	000023
20	000015	000024

	cOrderNo	ToyCost
1	000001	54.97
2	000002	86.50
3	000003	71.97
4	000004	35.99
5	000005	133.93
6	000006	53.97
7	000007	12.99
8	000008	31.98
9	000009	16.99
10	000010	43.97
11	000011	43.96
12	000012	14.99
13	000013	16.99
14	000014	31.98
15	000015	25.99
16	000016	0.00
17	000017	29.98
18	000018	29.98
19	000019	103.96
20	000020	0.00

16. 查找从来没有下过订单的顾客。

```
select cShopperId
from Shopper
```

```
where cShopperId not in(
    select cShopperId
    from Orders
)
```



17. 删除“Largo”牌的所有玩具。

具体思路：

如果要想删除相关的数据的话，会遇到因为由于外键的约束所以无法删除等情况，所以在这里，我选择先禁用外键约束，删除了“Largo”牌的所有玩具和相关信息之后，再重新调用外键约束。

```
-- 禁用外键约束
ALTER TABLE ShoppingCart NOCHECK CONSTRAINT
FK__ShoppingC__cToyI__4CA06362;
-- 删除 ShoppingCart 表中与 Largo 品牌关联的记录
delete from ShoppingCart
where cToyId IN (
    SELECT cToyId
    FROM Toys
    WHERE cBrandId IN (
        SELECT cBrandId
        FROM ToyBrand
        WHERE cBrandName = 'Largo'
    )
);
--删除 OrderDetail 表中与 Largo 相关记录
delete from OrderDetail
where cToyId in(
    select cToyId
    from Toys
    where cBrandId IN(
        SELECT cBrandId
        FROM ToyBrand
        WHERE cBrandName = 'Largo'
    )
```

```
)
--删除 PickOfMonth 表中与 Largo 相关记录
delete from PickOfMonth
where cToyId in(
	select cToyId
	from Toys
	where cBrandId IN(
		SELECT cBrandId
		FROM ToyBrand
		WHERE cBrandName = 'Largo'
	)
)

-- 启用外键约束
ALTER TABLE ShoppingCart CHECK CONSTRAINT FK__ShoppingC__cToyI__4CA06362;
```

```
(0 行受影响)

(136 行受影响)

(12 行受影响)

(4 行受影响)
```

实验报告要求：

1. 列出所有的 SQL 语句和源代码；
2. 程序要求有适当的注释；
3. 实验报告提交电子档。

实验内容（一）：存储过程与触发器

1. 创建一个称为 prcCharges 的存储过程，它返回某个定单号的装运费用和包装费用。

```
create procedure prcCharges
    @orderNo char(6),@mShippingCharges money output, @mGiftWrapCharges money output
as
begin
    select mShippingCharges, mGiftWrapCharges
    from Orders
    where @orderNo = Orders.cOrderNo
end
Go
```

2. 创建一个称为 prcHandlingCharges 的过程，它接收定单号并显示经营费用。PrchandlingCharges 过程应使用 prcCharges 过程来得到装运费和礼品包装费。提示：经营费用=装运费+礼品包装费

```
create procedure prcHandlingCharges
    @oNo char(6), @handleCharges money output
as
begin
    declare @shippingCharges money
    declare @giftWrapCharges money

    --调用 preCharges 存储过程  exec
    exec prcCharges @orderNo = @oNo, @mShippingCharges = @shippingCharges out,
@mGiftWrapCharges = @giftWrapCharges out
    set @handleCharges = @shippingCharges + @giftWrapCharges
    --返回参数
    select @handleCharges as HandlingCharges

end
go
```

3. 在 OrderDetail 上定义一个触发器，当向 OrderDetail 表中新增一条记录时，自动修改 Toys 表中玩具的库存数量（siToyQoh）。

```
create trigger OrderDetail_Insert
on OrderDetail
after insert
as
begin
    update toys
    set siToyQoh = siToyQoh - OrderDetail.siQty    --减去库存玩具数量
    from Toys,OrderDetail
    where Toys.cToyId = OrderDetail.cToyId
end
Go
```

4. Orders 表是 GlobalToyz 数据库里的一张核心的表，对这张表上做的任何更新动作（增、删、改）都需要记录下来，这是数据库审计（Audit）的基本思想。要求设计一张表存储对 Orders 表的更新操作，包括操作者、操作时间、操作类型、更新前的数据、更新后的数据。设计触发器实现对 Orders 表的审计。

具体思路：

（1） 使用 CREATE TABLE 语句创建一个名为 OrderAudit 的表，该表用于存储审计记录。该表包含以下列：

AuditID：审计记录的唯一标识，使用 IDENTITY(1,1)设置为自增长主键。

OrderID：被更新的订单的唯一标识。

ActionType：操作类型，例如插入（INSERT）、删除（DELETE）或更新（UPDATE）。

ActionDate：操作发生的日期和时间，使用 GETDATE()函数获取当前日期和时间。

UserID：执行操作的用户标识。

old_data：存储被更新记录的旧数据，使用 XML 数据类型存储。

new_data：存储被更新记录的新数据，使用 XML 数据类型存储。

（2） 使用 CREATE TRIGGER 语句创建一个名为 trg_OrderAudit 的触发器。该触发器将在 Orders 表上进行插入、更新和删除操作后触发。

（3） 在触发器的 BEGIN 和 END 之间定义触发器的操作逻辑。

- 插入操作：如果在插入操作后存在被插入的记录（通过检查 inserted 表是否存在记录），则将插入的记录的相关信息（如订单号、操作类型、操作日期、用户标识）以及 NULL 的值（因为是插入操作，没有旧数据）插入到 OrderAudit 表中。
- 更新操作：如果在更新操作后存在被删除的记录（通过检查 deleted 表是否存在记录），则将被删除的记录的相关信息（如订单号、操作类型、操作日期、用户标识）以及被更新的记录的相关信息（如新数据）插入到 OrderAudit 表中。
- 删除操作：如果在删除操作后存在被删除的记录（通过检查 deleted 表是否存在记录），则将被删除的记录的相关信息（如订单号、操作类型、操作日期、用户标识）以及 NULL 的值（因为是删除操作，没有新数据）插入到 OrderAudit 表中。

```
CREATE TABLE OrderAudit (
```

```
    AuditID INT IDENTITY(1,1) PRIMARY KEY, -- 审计记录的唯一标识（主键）。
    OrderID CHAR(6),-- 被更新的订单的唯一标识。
    ActionType VARCHAR(10),--操作类型，例如插入、删除或更新。
    ActionDate DATETIME,--操作发生的日期和时间。
    UserID VARCHAR(50),--执行操作的用户标识。
    old_data XML,
    new_data XML
);

CREATE TRIGGER trg_OrderAudit
ON Orders
AFTER INSERT, UPDATE, DELETE
AS
BEGIN
    -- 插入操作
    --FOR XML AUTO 指定将结果集转换为 XML 时使用自动元素名称
    IF EXISTS (SELECT * FROM inserted)
    BEGIN
        INSERT INTO OrderAudit (OrderID, ActionType, ActionDate, UserID, old_data,
new_data)
        SELECT i.cOrderNo, 'INSERT', GETDATE(), CURRENT_USER, (SELECT * FROM
inserted FOR XML AUTO), NULL
        FROM inserted i;
    END;

    -- 更新操作
    IF EXISTS (SELECT * FROM deleted)
    BEGIN
        INSERT INTO OrderAudit (OrderID, ActionType, ActionDate, UserID, old_data,
new_data)
        SELECT d.cOrderNo, 'UPDATE', GETDATE(), CURRENT_USER, (SELECT * FROM
deleted FOR XML AUTO), (SELECT * FROM inserted FOR XML AUTO)
        FROM deleted d
        JOIN inserted i ON d.cOrderNo = i.cOrderNo;
    END;

    -- 删除操作
    IF EXISTS (SELECT * FROM deleted)
    BEGIN
        INSERT INTO OrderAudit (OrderID, ActionType, ActionDate, UserID, old_data,
new_data)
        SELECT d.cOrderNo, 'DELETE', GETDATE(), CURRENT_USER, (SELECT * FROM
deleted FOR XML AUTO), NULL
        FROM deleted d;
```

```
    END;

END;
```

5. 编写代码，分析玩具和地域的关系，例如哪个城市的购买者对哪一种、哪一类或哪一个品牌的玩具更有兴趣。这道题是个开放的题目，同学们可以按照自己的理解从不同的角度进行分析。实验报告中需给出代码、结果截图和对分析结果的文字描述。

具体实现：

（1）首先是在 serversql 中选取数据做出一张新的表格：里面存储的是每个城市与品牌间的关系。

```
CREATE TABLE Toys_City (
    cBrandName char(20),
    cCategory char(20),
    vToyName nchar(20),
    cCity char(15)
);

INSERT INTO Toys_City (cBrandName, cCategory, vToyName, cCity)
SELECT ToyBrand.cBrandName,Category.cCategory ,Toys.vToyName, Shopper.cCity
FROM Orders
JOIN OrderDetail ON Orders.cOrderNo = OrderDetail.cOrderNo
JOIN Shopper ON Orders.cShopperId = Shopper.cShopperId
JOIN Toys ON OrderDetail.cToyId = Toys.cToyId
JOIN ToyBrand ON Toys.cBrandId = ToyBrand.cBrandId
JOIN Category ON Toys.cCategoryId = Category.cCategoryId;

--城市与玩具品牌的关系
CREATE TABLE City_Brand (
   cCity CHAR(15),
   cBrandName CHAR(20),
   num INT,
   PRIMARY KEY (cCity, cBrandName)
);

INSERT INTO City_Brand (cCity, cBrandName, num)
SELECT cCity, cBrandName, COUNT(*) AS num
FROM Toys_City
GROUP BY cCity, cBrandName
ORDER BY cCity, num DESC;
```

（2）然后在 pycham 中连接上 serversql，通过新表的数据使用 matplotlib 来进行绘图及数据分析：

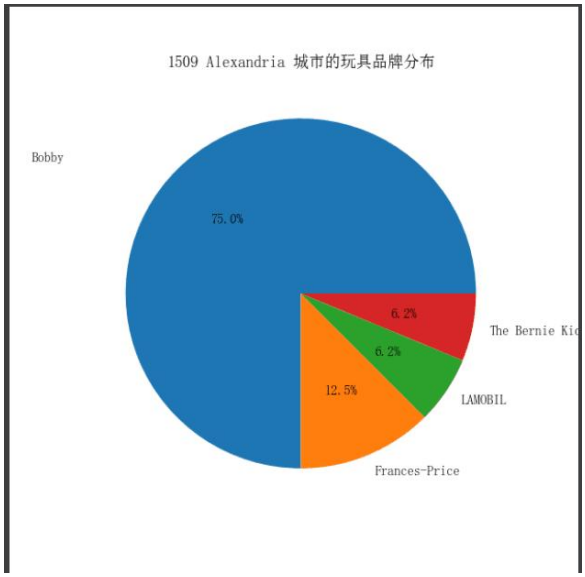
```
import pymssql

# 直接找管理员权限！
from matplotlib import pyplot as plt, cm

connect = pymssql.connect(host='localhost', server='LAPTOP-IS2KHG9B', port='1433', user='sa',
password='123456',
                                    database='GlobalToyz')

if connect:
    print("数据库连接成功")
else:
    print("连接失败")

cursor = connect.cursor()
sqlQuery = "select * from City_Brand"

results = []
try:
    cursor.execute(sqlQuery)
    results = cursor.fetchall()
    print(results)
    print(len(results))   # 多少行数据
except Exception as e:
    print(e)

connect.close()

# 城市所对应的玩具品牌名和数量
citys = []
city_brand_dict = {}
for result in results:
    city = result[0]
    brand = result[1]
    count = result[2]
    if city not in city_brand_dict:
        city_brand_dict[city] = {}
        citys.append(city)
    if brand not in city_brand_dict[city]:
        city_brand_dict[city][brand] = count
    else:
        city_brand_dict[city][brand] += count

print(city_brand_dict)
```

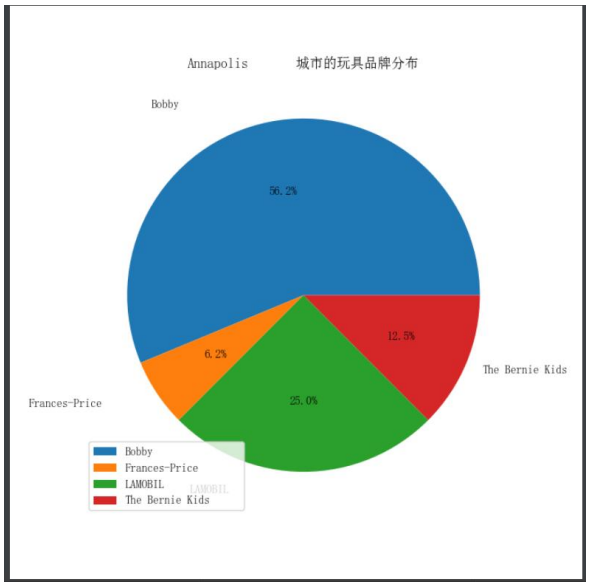
```
print(citys)

# 获取第一个城市所对应的玩具品牌名和数量
city, brands_dict = None, {}
if results:
    city = results[0][0]
for row in results:
    if row[0] != city:
        break
    brand, count = row[1], row[2]
    brands_dict[brand] = count

# 绘制扇形图
plt.rcParams['font.family'] = ['SimSun']
fig, ax = plt.subplots(figsize=(6, 6))
ax.pie(brands_dict.values(), labels=brands_dict.keys(), autopct='%1.1f%%')
ax.set_title(f'{city} 城市的玩具品牌分布')
plt.legend()
plt.show()

# 获取第二个城市所对应的玩具品牌名和数量
brands_dict2 = {}
city2 = citys[1]
for row in results:
    if row[0] == city:    # 说明是第一个城市的数据
        continue
    elif row[0] != city2:
        break
    brand, count = row[1], row[2]
    brands_dict2[brand] = count

# 绘制扇形图
fig, ax = plt.subplots(figsize=(7, 7))
ax.pie(brands_dict2.values(), labels=brands_dict2.keys(), autopct='%1.1f%%')
ax.set_title(f'{city2} 城市的玩具品牌分布')
plt.legend()
plt.show()


# 获取第三个城市所对应的玩具品牌名和数量
brands_dict3 = {}
city3 = citys[2]
for row in results:
    if row[0] == city:    # 说明是第一个城市的数据
```

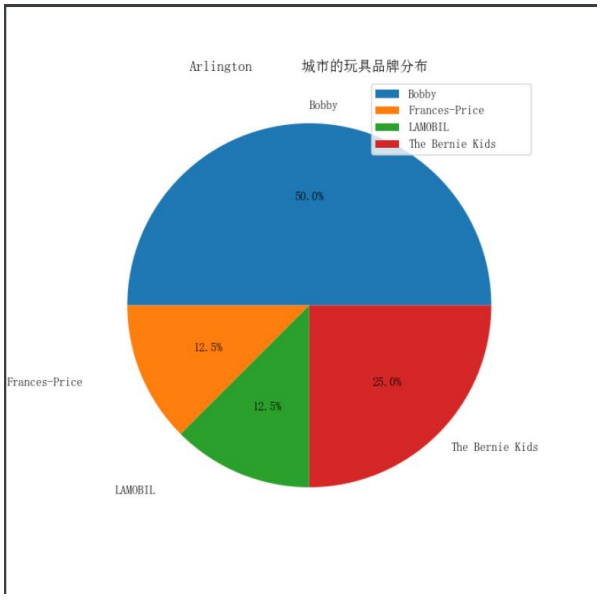
```
            continue
        elif row[0] == city2:
            continue
        elif row[0] != city3:
            break
        brand, count = row[1], row[2]
        brands_dict3[brand] = count

# 绘制扇形图
fig, ax = plt.subplots(figsize=(7, 7))
ax.pie(brands_dict3.values(), labels=brands_dict3.keys(), autopct='%1.1f%%')
ax.set_title(f'{city3} 城市的玩具品牌分布')
plt.legend()
plt.show()


# 获取第四个城市所对应的玩具品牌名和数量
brands_dict4 = {}
city4 = citys[3]
for row in results:
    if row[0] == city:    # 说明是第一个城市的数据
        continue
    elif row[0] == city2:
        continue
    elif row[0] == city3:
        continue
    elif row[0] != city4:
        break
    brand, count = row[1], row[2]
    brands_dict4[brand] = count

# 绘制扇形图
fig, ax = plt.subplots(figsize=(7, 7))
ax.pie(brands_dict4.values(), labels=brands_dict4.keys(), autopct='%1.1f%%')
ax.set_title(f'{city4} 城市的玩具品牌分布')
plt.legend()
plt.show()


# 获取第五个城市所对应的玩具品牌名和数量
brands_dict5 = {}
city5 = citys[4]
for row in results:
    if row[0] == city:    # 说明是第一个城市的数据
```

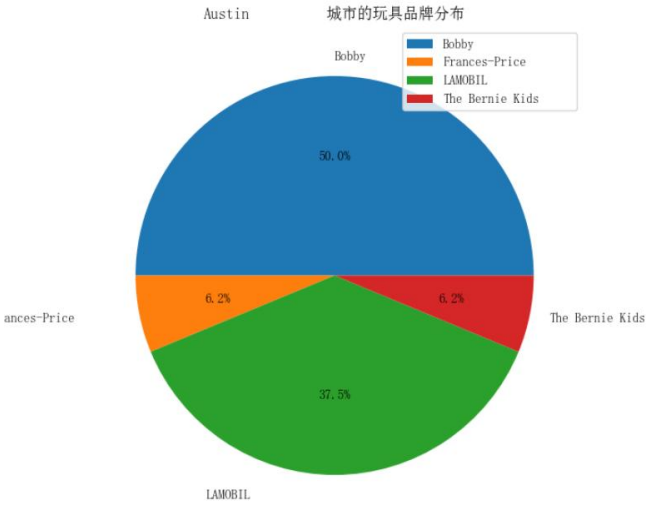
```
            continue
        elif row[0] == city2:
            continue
        elif row[0] == city3:
            continue
        elif row[0] == city4:
            continue
        elif row[0] != city5:
            break
        brand, count = row[1], row[2]
        brands_dict5[brand] = count

# 绘制扇形图
fig, ax = plt.subplots(figsize=(7, 7))
ax.pie(brands_dict5.values(), labels=brands_dict5.keys(), autopct='%1.1f%%')
ax.set_title(f'{city5} 城市的玩具品牌分布')
plt.legend()
plt.show()
```

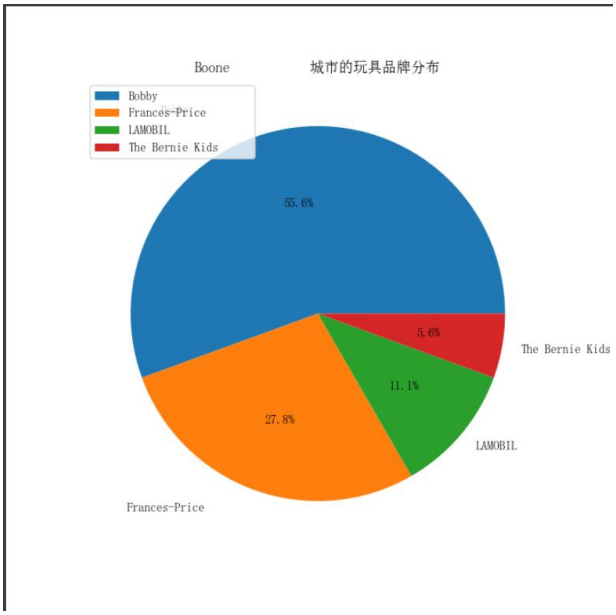
（3）这里只分析了前五个城市与玩具品牌间的关系，绘图如下：从图中也可看出每个城市中玩具品牌受欢迎的比例。




Annapolis 城市的玩具品牌分布
Bobby
56.2%
12.5%
6.2%
The Bernie Kids
Frances-Price
25.0%
Bobby
Frances-Price
LAMOBIL
The Bernie Kids


Arlington 城市的玩具品牌分布
Bobby
Bobby
Frances-Price
LAMOBIL
The Bernie Kids
50.0%
12.5%
Frances-Price
25.0%
12.5%
The Bernie Kids
LAMOBIL


Austin 城市的玩具品牌分布
Bobby
Bobby
Frances-Price
LAMOBIL
The Bernie Kids
50.0%
6.2%
6.2%
ances-Price
The Bernie Kids
37.5%
LAMOBIL



实验内容（二）：视图、事务与游标

1. 定义一个视图，包括定单的编号、时间、金额以及收货人的姓名、国家代码和国家名称。

```
create view OrderView as
select o.cOrderNo, dOrderDate, mTotalCost,vFirstName, vLastName,
cState,cCountryId
from Orders o, Recipient r
where o.cOrderNo = r.cOrderNo
go
```

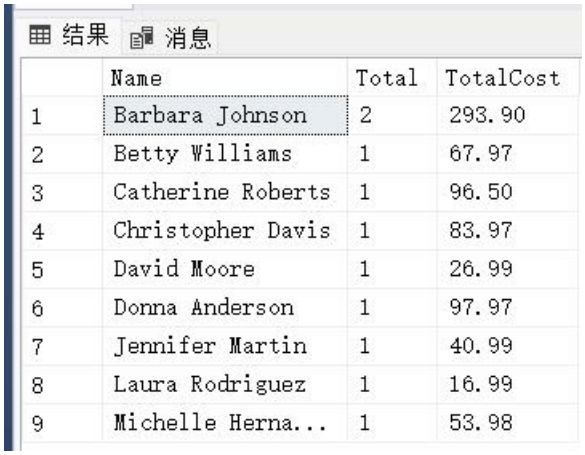
LAPTOP-IS2KHG9...- dbo.OrderView | LAPTOP-IS2KHG9B....- dbo.Recipient | LAPTOP-IS2KH

	cOrderNo	dOrderDa...	mTotalCost	vFirstName	vLastName	cState	cCountryId
▶	000001	2017-05-2...	62.2200	Barbara	Johnson	California	001
	000002	2017-05-2...	96.5000	Catherine	Roberts	California	001
	000003	2017-05-2...	83.9700	Christopher	Davis	Utah	001
	000004	2017-05-2...	40.9900	Jennifer	Martin	Maryland ...	001
	000005	2017-05-2...	231.6800	Barbara	Johnson	California	001
	000006	2017-05-2...	97.9700	Donna	Anderson	Nevada ...	001
	000007	2017-05-2...	16.9900	Laura	Rodriguez	Georgia	001
	000008	2017-05-2...	53.9800	Michelle	Hernandez	Illinois	001
	000009	2017-05-2...	26.9900	David	Moore	California	001
	000010	2017-05-2...	67.9700	Betty	Williams	Virginia	001
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

2. 基于（1）中定义的视图，查询所有国家代码为‘001’的收货人的姓名和他们所下定单的笔数及定单的总金额。

```
select (vFirstName +' '+ vLastName) as Name, COUNT(cOrderNo) as Total,
SUM(mTotalCost) as TotalCost
```

```
from OrderView
where cCountryId = '001'
group by (vFirstName + ' '+ vLastName)
go
```



结果 消息

	Name	Total	TotalCost
1	Barbara Johnson	2	293.90
2	Betty Williams	1	67.97
3	Catherine Roberts	1	96.50
4	Christopher Davis	1	83.97
5	David Moore	1	26.99
6	Donna Anderson	1	97.97
7	Jennifer Martin	1	40.99
8	Laura Rodriguez	1	16.99
9	Michelle Herna...	1	53.98

3. 视图定义如下：

```
CREATE VIEW vwOrderWrapper
    AS
    SELECT cOrderNo, cToyId, siQty, vDescription, mWrapperRate
    FROM OrderDetail JOIN Wrapper
    ON OrderDetail.cWrapperId = Wrapper.cWrapperId
```

执行以下更新命令并分析该命令的执行结果。

```
UPDATE vwOrderWrapper
SET siQty = 2, mWrapperRate = mWrapperRate + 1
WHERE cOrderNo = '000001'
```

```
CREATE VIEW vwOrderWrapper
AS
    SELECT cOrderNo, cToyId, siQty, vDescription, mWrapperRate
    FROM OrderDetail JOIN Wrapper
    ON OrderDetail.cWrapperId = Wrapper.cWrapperId
go
```

cOrderNo	cToyId	siQty	vDescripti...	mWrappe...
000001	000008	1	Baby blocks	1.2500
000002	000016	2	Geckos	1.0000
000004	000030	1	Geckos	1.0000
000005	000001	4	Geckos	1.0000
000005	000024	1	Baby blocks	1.2500
000005	000030	2	Baby blocks	1.2500
000006	000013	2	Stars	1.5000
000006	000017	1	Geckos	1.0000
000008	000023	2	Geckos	1.0000
000009	000018	1	Bubbles	2.0000
000010	000020	2	Sesame st...	1.5000
000011	000012	4	Geckos	1.0000
000012	000032	1	Sesame st...	1.5000
000013	000018	1	Baby blocks	1.2500
000016	000016	0	Moon	2.2500
000017	000032	2	Sesame st...	1.5000
000018	000032	2	Moon	2.2500
000021	000008	3	Sea	1.0000
000025	000017	1	Sesame st...	1.5000
000026	000016	3	Sesame st...	1.5000
000028	000007	3	Moon	2.2500
000030	000001	0	Sesame st...	1.5000

```
25
26  UPDATE vwOrderWrapper
27    SET siQty = 2, mWrapperRate = mWrapperRate + 1
28    WHERE cOrderNo = '000001'
29
30
```

160 %

消息

消息 4405，级别 16，状态 1，第 26 行
视图或函数 'vwOrderWrapper' 不可更新，因为修改会影响多个基表。

分析：视图中的数据来自多个基表，无法直接进行更新；
要更新的话，需要找到所有的基表，然后分别更新它们。

```
--instead of 触发器
create trigger Wrapper_update
on vwOrderWrapper
instead of update
as
begin
    --更新基表 OrderDetail
    update OrderDetail
    set siQty = i.siQty, cWrapperId = w.cWrapperId
    from inserted i join OrderDetail od
    on i.cOrderNo = od.cOrderNo and i.cToyId = od.cToyId
    join Wrapper w
```

```
    on i.mWrapperRate = w.mWrapperRate

    --更新基表 Wrapper
    update Wrapper
    set mWrapperRate = i.mWrapperRate
    from inserted i join OrderDetail od
    on i.cOrderNo = od.cOrderNo
    join Wrapper w
    on od.cWrapperId = w.cWrapperId
end

UPDATE vwOrderWrapper
SET siQty = 2, mWrapperRate = mWrapperRate + 1
WHERE cOrderNo = '000001'
```

注意：在这里，"Instead of" 触发器不会对数据库中的数据产生影响。当 "Instead of" 触发器被激活时，它会代替原始操作并执行定义在触发器中的逻辑

4. 在 GlobalToyz 数据库里创建一个用户，用户名为 user_xxxx（你的学号）。通过视图限制该用户只能访问 Orders 表中 2017 年以前的数据。

注意：由于我的表中现在没有 2017 年以前的数据，所以这里更改为 2018 年前。

```
create login user_001 with password = '123145'
go

create user user_001 for login user_001
go

--创建一个视图，只存放 2018 年以前的数据
create view Orders_2018
as
select *
from Orders
where dOrderDate < '2018-01-01 00:00:00.000'
```

```
go
--授权
grant select on Orders_2018 to user_001;
go
```

5. 当购物者确认定单时，应该包含下面的步骤：
   （1）产生新的定单号（要求创建一个存储过程，用于产生新定单号）。
   （2）定单号，当前日期，购物车 ID，和购物者 ID 应该加到 Orders 表中。
   （3）定单号，玩具 ID 和数量应加到 OrderDetail 表中。
   （4）在 OrderDetail 表中更新玩具成本。（提示：Toy cost = Quantity * Toy Rate）。
   （5）从 ShoppingCart 表中将本次已购买的玩具删除。
   将上述步骤定义为一个事务。编写一个过程以购物车 ID 和购物者 ID 为参数，实现这个事务。（提示：首先需要修改表 ShoppingCart 的结构，在表中新增一个字段‘Status’。该字段取值为 1，表示该玩具为本次下订单时要购买的玩具，并产生一些模拟数据。）

```
--1).首先需要修改表 ShoppingCart 的结构，在表中新增一个字段‘Status’
--修改表 ShoppingCart 结构
alter table ShoppingCart
add Status smallint

update ShoppingCart
set Status = 1
go
--2).创建名为 prcGenOrder 的存储过程，产生存在于数据库中的定单号
--生成新的订单号
create procedure prcGenOrder
@OrderNo char(6) output
as
    select @OrderNo=Max(cOrderNo)
    from Orders

    select @OrderNo=
    case
        when @OrderNo>=0 and @OrderNo<9 Then
                '00000'+Convert(char,@OrderNo+1)
        when @OrderNo>=9 and @OrderNo<99 Then
                '0000'+Convert(char,@OrderNo+1)
        when @OrderNo>=99 and @OrderNo<999 Then
                '000'+Convert(char,@OrderNo+1)
        when @OrderNo>=999 and @OrderNo<9999 Then
                '00'+Convert(char,@OrderNo+1)
        when @OrderNo>=9999 and @OrderNo<99999 Then
                '0'+Convert(char,@OrderNo+1)
```

```
        when @OrderNo>=99999 Then
                Convert(char,@OrderNo+1)
    end

    print @OrderNo
return
Go
--3).定义为一个事务，编写一个过程以购物车 ID 和购物者 ID 为参数，实现这个事务。
--    定单号，当前日期，购物车 ID，和购物者 ID 应该加到 Orders 表中。
--    定单号，玩具 ID 和数量应加到 OrderDetail 表中。
--    在 OrderDetail 表中更新玩具成本。（提示：Toy cost = Quantity * Toy Rate）。
--    从 ShoppingCart 表中将本次已购买的玩具删除。

begin transaction
    --count 记录事务中出现错误的次数
    declare @count int
    set @count=0
    declare @OrderNo char(6)
    exec prcGenOrder @OrderNo output
    set @count=@count+@@ERROR

    declare @ToyId char (6)
    declare @ShopperId char (6)
    declare @ToyRate money
    declare @CartId char (6)
    declare @Qty int
    set @CartId='000009'
    set @ShopperId='000007'
    set @ToyId='000008'
    set @Qty=2

    select @ToyRate=mToyRate
    from Toys
    where cToyId=@ToyId

    --定单号，当前日期，购物车 ID，和购物者 ID 应该加到 Orders 表中
    insert into Orders
    values(@OrderNo,getdate(),@CartId,@ShopperId,null,null,null,null,null,null)
    set @count=@count+@@ERROR --记录上条语句是否产生错误

    --定单号，玩具 ID 和数量应加到 OrderDetail 表中
    insert into OrderDetail
    values(@OrderNo,@ToyId,@Qty,null,null,null,@Qty*@ToyRate)
    set @count=@count+@@ERROR --记录上条语句是否产生错误
```

```
    --从 ShoppingCart 表中将本次已购买的玩具删除
    delete from ShoppingCart where Status=1 and cToyId=@ToyId
    set @count=@count+@@ERROR --记录上条语句是否产生错误

    --如果@count 的值大于 0，则说明事务中有一个或多个操作失败，此时需要进行回滚；否则，所有操作都成功执行，可以进行提交。
    if(@count>0)
    begin
        print'Error'
        rollback
    end
    else
        commit
go
```

消息

(1 行受影响)

(25 行受影响)

(1 行受影响)

(1 行受影响)

Orders 表：

1011	001011	2023-06-30 03:00:00.000	000002	000033	01	6.00	0.00	Y	77.97	2028-11-30 00:00:00.000
1012	001012	2023-06-01 20:57:52.090	000009	000007	NULL	NULL	NULL	NULL	NULL	NULL

OrderDetail 表：

881	001012	000008	2	NULL	NULL	NULL	29.98

6. 编写一个程序显示每天的定单状态。如果当天的定单值总合大于 150，则显示“High sales”,否则显示”Low sales”。要求列出日期、定单状态和定单总价值。（**要求用游标实现**）

具体思路：

（1） 首先，声明了几个变量：@OrderDate 用于存储订单日期，@OrderAmount 用于存储订单金额，@TotalSales 用于存储累计销售额，@OrderStatus 用于存储订单状态。

（2） 定义了一个名为 order_cursor 的游标，该游标从 Orders 表中选择订单日期和订单金额。

（3） 使用 OPEN 语句打开游标，并使用 FETCH NEXT 语句将游标定位到第一行数据，并将该行的日期和金额分别存储在@OrderDate 和@OrderAmount 变量中。

（4） 使用 WHILE 循环来遍历游标的每一行数据。在循环内部，将订单金额加到累计销售额变量@TotalSales 上。

（5） 使用 IF 条件语句根据累计销售额判断订单状态，如果超过 150 则设置订单状态为"High sales"，否则设置为"Low sales"。

（6） 使用 PRINT 语句打印出订单日期、订单状态和累计销售额。通过 CONVERT 函数将日期、销售额转换为字符串格式。

（7） 使用 FETCH NEXT 语句将游标移动到下一行数据，并将对应的日期和金额存储到相应的变量中。

（8） 在循环结束后，使用 CLOSE 语句关闭游标。

（9） 使用 DEALLOCATE 语句释放游标所占用的资源

```
DECLARE @OrderDate DATE
DECLARE @OrderAmount MONEY
DECLARE @TotalSales MONEY = 0
DECLARE @OrderStatus NVARCHAR(20)

--定义游标
DECLARE order_cursor CURSOR FOR
SELECT dOrderDate, mTotalCost
FROM Orders

--打开游标
OPEN order_cursor

FETCH NEXT FROM order_cursor INTO @OrderDate, @OrderAmount

WHILE @@FETCH_STATUS = 0
BEGIN
    SET @TotalSales = @TotalSales + @OrderAmount

    IF @TotalSales > 150
        SET @OrderStatus = 'High sales'
    ELSE
        SET @OrderStatus = 'Low sales'

    PRINT CONVERT(NVARCHAR(10), @OrderDate, 120) + ': ' + @OrderStatus + ', Total
Order Amount: ' + CONVERT(NVARCHAR(20), @TotalSales, 2)

    FETCH NEXT FROM order_cursor INTO @OrderDate, @OrderAmount
END

CLOSE order_cursor
--释放游标
DEALLOCATE order_cursor
```

```
2017-05-20: Low sales, Total Order Amount: 62.2200
2017-05-20: High sales, Total Order Amount: 158.7200
2017-05-20: High sales, Total Order Amount: 242.6900
2017-05-20: High sales, Total Order Amount: 283.6800
2017-05-21: High sales, Total Order Amount: 515.3600
2017-05-21: High sales, Total Order Amount: 613.3300
2017-05-22: High sales, Total Order Amount: 630.3200
2017-05-22: High sales, Total Order Amount: 684.3000
2017-05-22: High sales, Total Order Amount: 711.2900
2017-05-22: High sales, Total Order Amount: 779.2600
2023-05-19: High sales, Total Order Amount: 867.2200
2023-05-19: High sales, Total Order Amount: 885.7100
2023-05-19: High sales, Total Order Amount: 905.9500
2023-05-19: High sales, Total Order Amount: 941.9300
2023-05-19: High sales, Total Order Amount: 969.9200
```

7. 基于表 Orders 和 Shopper，以下列格式生成报表：

购货人 ID　XXX　购货人姓名　XXX

购货人地址　XXXXXX

定单号 XXX　定单时间 XXX　定单金额 XXX

定单号 XXX　定单时间 XXX　定单金额 XXX

```
DECLARE order_cursor CURSOR FOR
SELECT Shopper.cShopperId ,
       CONCAT(Shopper.vFirstName,' ',Shopper.vLastName) ,
       Shopper.vAddress ,
       Orders.cOrderNo ,
       Orders.dOrderDate ,
       Orders.mTotalCost
FROM Orders
INNER JOIN Shopper ON Orders.cShopperId = Shopper.cShopperId

declare @cShopperId char(6)
declare @cShopperName varchar(40)
declare @vAddress varchar(40)
declare @cOrderNo char(6)
declare @dOrderDate Datetime
declare @mTotalCost money

--打开游标
OPEN order_cursor

FETCH NEXT FROM order_cursor INTO @cShopperId, @cShopperName, @vAddress, @cOrderNo,
@dOrderDate, @mTotalCost
```

```
    WHILE @@FETCH_STATUS = 0
    BEGIN

        PRINT '购货人 ID' + CONVERT(NVARCHAR(6), @cShopperId) + '    ' +
        '购货人姓名 ' + CONVERT(NVARCHAR(40), @cShopperName) +
CHAR(13)+CHAR(10)+
        '购货人地址' + CONVERT(NVARCHAR(40), @vAddress) + CHAR(13)+CHAR(10)+
        '定单号'+ CONVERT(NVARCHAR(6), @cOrderNo)  + '    '+
        '定单时间' + CONVERT(NVARCHAR(20), @dOrderDate) + '    ' +
        '定单金额' + CONVERT(NVARCHAR(6), @mTotalCost)+ CHAR(13)+CHAR(10);

        FETCH NEXT FROM order_cursor INTO @cShopperId, @cShopperName, @vAddress,
@cOrderNo, @dOrderDate, @mTotalCost

    END

CLOSE order_cursor
```

```
购货人ID000002    购货人姓名 Barbara Johnson
购货人地址227 Beach Ave.
定单号000001    定单时间05 20 2017 12:00AM    定单金额62.22
购货人ID000005    购货人姓名 Catherine Roberts
购货人地址5508 Aquiline Court
定单号000002    定单时间05 20 2017 12:00AM    定单金额96.50
购货人ID000007    购货人姓名 Christopher Davis
购货人地址4896 11th ST
定单号000003    定单时间05 20 2017 12:00AM    定单金额83.97
购货人ID000006    购货人姓名 Charles Brown
购货人地址7822 S. Glitzy Avenue
定单号000004    定单时间05 20 2017 12:00AM    定单金额40.99
购货人ID000002    购货人姓名 Barbara Johnson
购货人地址227 Beach Ave.
定单号000005    定单时间05 21 2017 12:00AM    定单金额231.68
```