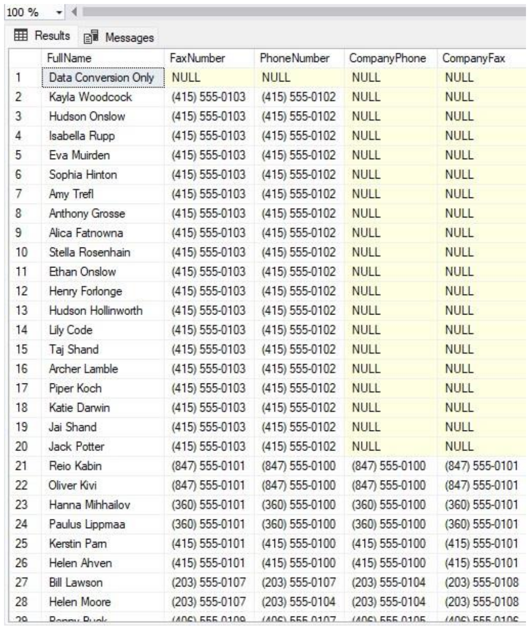
SQL Assignments

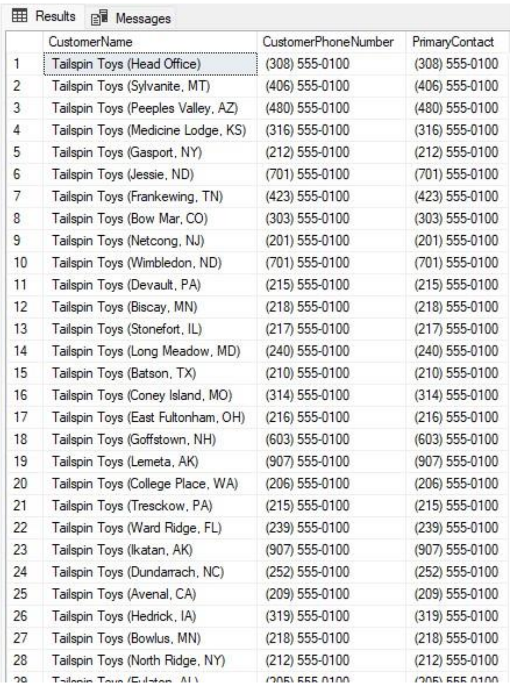
SQL related assignments will be on the Wide World Importers Database unless otherwise mentioned.

1. List of Persons' full name, all their fax and phone numbers, as well as the phone number and fax of the company they are working for (if any).



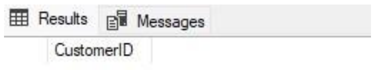
	FullName	FaxNumber	PhoneNumber	CompanyPhone	CompanyFax
1	Data Conversion Only	NULL	NULL	NULL	NULL
2	Kayla Woodcock	(415) 555-0103	(415) 555-0102	NULL	NULL
3	Hudson Onslow	(415) 555-0103	(415) 555-0102	NULL	NULL
4	Isabella Rupp	(415) 555-0103	(415) 555-0102	NULL	NULL
5	Eva Muirden	(415) 555-0103	(415) 555-0102	NULL	NULL
6	Sophia Hinton	(415) 555-0103	(415) 555-0102	NULL	NULL
7	Amy Trefl	(415) 555-0103	(415) 555-0102	NULL	NULL
8	Anthony Grosse	(415) 555-0103	(415) 555-0102	NULL	NULL
9	Alica Fatnowna	(415) 555-0103	(415) 555-0102	NULL	NULL
10	Stella Rosenhain	(415) 555-0103	(415) 555-0102	NULL	NULL
11	Ethan Onslow	(415) 555-0103	(415) 555-0102	NULL	NULL
12	Henry Forlonge	(415) 555-0103	(415) 555-0102	NULL	NULL
13	Hudson Hollinworth	(415) 555-0103	(415) 555-0102	NULL	NULL
14	Lily Code	(415) 555-0103	(415) 555-0102	NULL	NULL
15	Taj Shand	(415) 555-0103	(415) 555-0102	NULL	NULL
16	Archer Lamble	(415) 555-0103	(415) 555-0102	NULL	NULL
17	Piper Koch	(415) 555-0103	(415) 555-0102	NULL	NULL
18	Katie Darwin	(415) 555-0103	(415) 555-0102	NULL	NULL
19	Jai Shand	(415) 555-0103	(415) 555-0102	NULL	NULL
20	Jack Potter	(415) 555-0103	(415) 555-0102	NULL	NULL
21	Reio Kabin	(847) 555-0101	(847) 555-0100	(847) 555-0100	(847) 555-0101
22	Oliver Kivi	(847) 555-0101	(847) 555-0100	(847) 555-0100	(847) 555-0101
23	Hanna Mihhailov	(360) 555-0101	(360) 555-0100	(360) 555-0100	(360) 555-0101
24	Paulus Lippmaa	(360) 555-0101	(360) 555-0100	(360) 555-0100	(360) 555-0101
25	Kerstin Parn	(415) 555-0101	(415) 555-0100	(415) 555-0100	(415) 555-0101
26	Helen Ahven	(415) 555-0101	(415) 555-0100	(415) 555-0100	(415) 555-0101
27	Bill Lawson	(203) 555-0107	(203) 555-0107	(203) 555-0104	(203) 555-0108
28	Helen Moore	(203) 555-0107	(203) 555-0104	(203) 555-0104	(203) 555-0108

2. If the customer's primary contact person has the same phone number as the customer's phone number, list the customer companies.



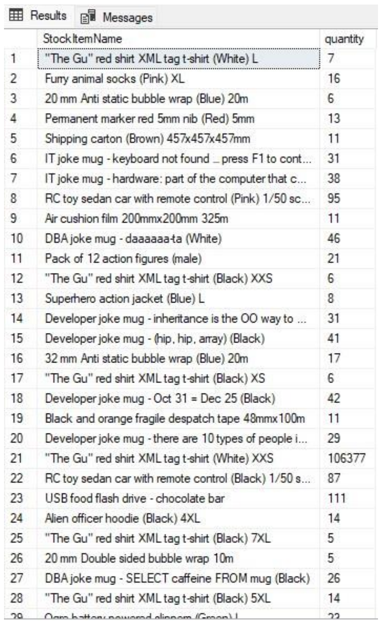
	CustomerName	CustomerPhoneNumber	PrimaryContact
1	Tailspin Toys (Head Office)	(308) 555-0100	(308) 555-0100
2	Tailspin Toys (Sylvanite, MT)	(406) 555-0100	(406) 555-0100
3	Tailspin Toys (Peeples Valley, AZ)	(480) 555-0100	(480) 555-0100
4	Tailspin Toys (Medicine Lodge, KS)	(316) 555-0100	(316) 555-0100
5	Tailspin Toys (Gasport, NY)	(212) 555-0100	(212) 555-0100
6	Tailspin Toys (Jessie, ND)	(701) 555-0100	(701) 555-0100
7	Tailspin Toys (Frankewing, TN)	(423) 555-0100	(423) 555-0100
8	Tailspin Toys (Bow Mar, CO)	(303) 555-0100	(303) 555-0100
9	Tailspin Toys (Netcong, NJ)	(201) 555-0100	(201) 555-0100
10	Tailspin Toys (Wimbledon, ND)	(701) 555-0100	(701) 555-0100
11	Tailspin Toys (Devault, PA)	(215) 555-0100	(215) 555-0100
12	Tailspin Toys (Biscay, MN)	(218) 555-0100	(218) 555-0100
13	Tailspin Toys (Stonefort, IL)	(217) 555-0100	(217) 555-0100
14	Tailspin Toys (Long Meadow, MD)	(240) 555-0100	(240) 555-0100
15	Tailspin Toys (Batson, TX)	(210) 555-0100	(210) 555-0100
16	Tailspin Toys (Coney Island, MO)	(314) 555-0100	(314) 555-0100
17	Tailspin Toys (East Fultonham, OH)	(216) 555-0100	(216) 555-0100
18	Tailspin Toys (Goffstown, NH)	(603) 555-0100	(603) 555-0100
19	Tailspin Toys (Lemeta, AK)	(907) 555-0100	(907) 555-0100
20	Tailspin Toys (College Place, WA)	(206) 555-0100	(206) 555-0100
21	Tailspin Toys (Tresckow, PA)	(215) 555-0100	(215) 555-0100
22	Tailspin Toys (Ward Ridge, FL)	(239) 555-0100	(239) 555-0100
23	Tailspin Toys (Ikatan, AK)	(907) 555-0100	(907) 555-0100
24	Tailspin Toys (Dundarrach, NC)	(252) 555-0100	(252) 555-0100
25	Tailspin Toys (Avenal, CA)	(209) 555-0100	(209) 555-0100
26	Tailspin Toys (Hedrick, IA)	(319) 555-0100	(319) 555-0100
27	Tailspin Toys (Bowlus, MN)	(218) 555-0100	(218) 555-0100
28	Tailspin Toys (North Ridge, NY)	(212) 555-0100	(212) 555-0100

3. List of customers to whom we made a sale prior to 2016 but no sale since 2016-01-01.



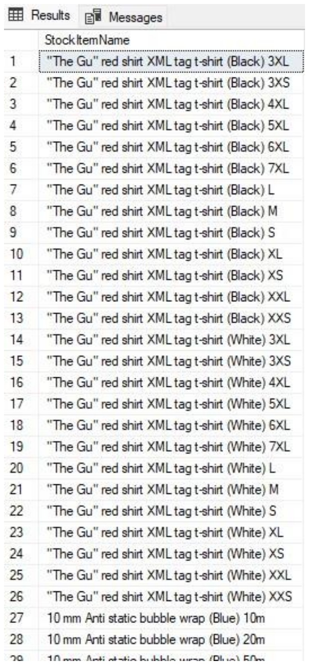
CustomerID

4. List of Stock Items and total quantity for each stock item in Purchase Orders in Year 2013.



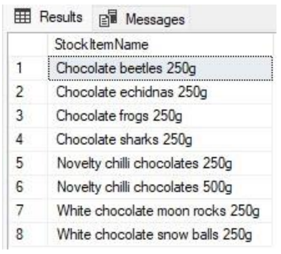
	StockItemName	quantity
1	"The Gu" red shirt XML tag t-shirt (White) L	7
2	Furry animal socks (Pink) XL	16
3	20 mm Anti static bubble wrap (Blue) 20m	6
4	Permanent marker red 5mm nib (Red) 5mm	13
5	Shipping carton (Brown) 457x457x457mm	11
6	IT joke mug - keyboard not found ... press F1 to cont...	31
7	IT joke mug - hardware: part of the computer that c...	38
8	RC toy sedan car with remote control (Pink) 1/50 sc...	95
9	Air cushion film 200mmx200mm 325m	11
10	DBA joke mug - daaaaaa-ta (White)	46
11	Pack of 12 action figures (male)	21
12	"The Gu" red shirt XML tag t-shirt (Black) XXS	6
13	Superhero action jacket (Blue) L	8
14	Developer joke mug - inheritance is the OO way to ...	31
15	Developer joke mug - (hip, hip, array) (Black)	41
16	32 mm Anti static bubble wrap (Blue) 20m	17
17	"The Gu" red shirt XML tag t-shirt (Black) XS	6
18	Developer joke mug - Oct 31 = Dec 25 (Black)	42
19	Black and orange fragile despatch tape 48mmx100m	11
20	Developer joke mug - there are 10 types of people i...	29
21	"The Gu" red shirt XML tag t-shirt (White) XXS	106377
22	RC toy sedan car with remote control (Black) 1/50 s...	87
23	USB food flash drive - chocolate bar	111
24	Alien officer hoodie (Black) 4XL	14
25	"The Gu" red shirt XML tag t-shirt (Black) 7XL	5
26	20 mm Double sided bubble wrap 10m	5
27	DBA joke mug - SELECT caffeine FROM mug (Black)	26
28	"The Gu" red shirt XML tag t-shirt (Black) 5XL	14

5. List of stock items that have at least 10 characters in description.



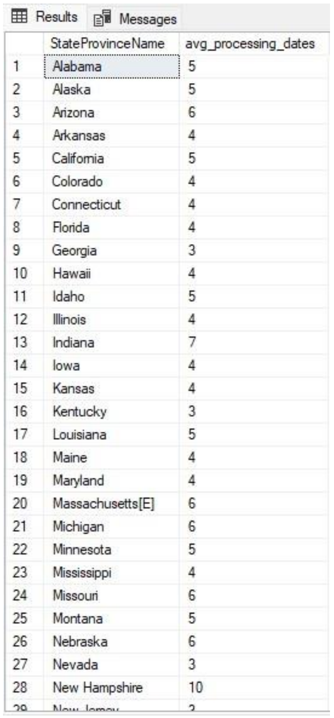
	StockItemName
1	"The Gu" red shirt XML tag t-shirt (Black) 3XL
2	"The Gu" red shirt XML tag t-shirt (Black) 3XS
3	"The Gu" red shirt XML tag t-shirt (Black) 4XL
4	"The Gu" red shirt XML tag t-shirt (Black) 5XL
5	"The Gu" red shirt XML tag t-shirt (Black) 6XL
6	"The Gu" red shirt XML tag t-shirt (Black) 7XL
7	"The Gu" red shirt XML tag t-shirt (Black) L
8	"The Gu" red shirt XML tag t-shirt (Black) M
9	"The Gu" red shirt XML tag t-shirt (Black) S
10	"The Gu" red shirt XML tag t-shirt (Black) XL
11	"The Gu" red shirt XML tag t-shirt (Black) XS
12	"The Gu" red shirt XML tag t-shirt (Black) XXL
13	"The Gu" red shirt XML tag t-shirt (Black) XXS
14	"The Gu" red shirt XML tag t-shirt (White) 3XL
15	"The Gu" red shirt XML tag t-shirt (White) 3XS
16	"The Gu" red shirt XML tag t-shirt (White) 4XL
17	"The Gu" red shirt XML tag t-shirt (White) 5XL
18	"The Gu" red shirt XML tag t-shirt (White) 6XL
19	"The Gu" red shirt XML tag t-shirt (White) 7XL
20	"The Gu" red shirt XML tag t-shirt (White) L
21	"The Gu" red shirt XML tag t-shirt (White) M
22	"The Gu" red shirt XML tag t-shirt (White) S
23	"The Gu" red shirt XML tag t-shirt (White) XL
24	"The Gu" red shirt XML tag t-shirt (White) XS
25	"The Gu" red shirt XML tag t-shirt (White) XXL
26	"The Gu" red shirt XML tag t-shirt (White) XXS
27	10 mm Anti static bubble wrap (Blue) 10m
28	10 mm Anti static bubble wrap (Blue) 20m

6. List of stock items that are not sold to the state of Alabama and Georgia in 2014.



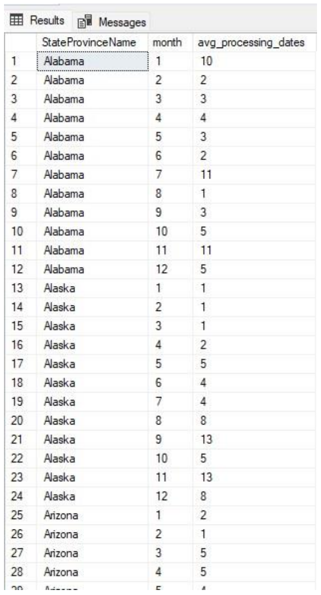
	StockItemName
1	Chocolate beetles 250g
2	Chocolate echidnas 250g
3	Chocolate frogs 250g
4	Chocolate sharks 250g
5	Novelty chilli chocolates 250g
6	Novelty chilli chocolates 500g
7	White chocolate moon rocks 250g
8	White chocolate snow balls 250g

7. List of States and Avg dates for processing (confirmed delivery date – order date).



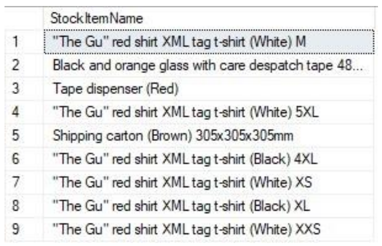
	StateProvinceName	avg_processing_dates
1	Alabama	5
2	Alaska	5
3	Arizona	6
4	Arkansas	4
5	California	5
6	Colorado	4
7	Connecticut	4
8	Florida	4
9	Georgia	3
10	Hawaii	4
11	Idaho	5
12	Illinois	4
13	Indiana	7
14	Iowa	4
15	Kansas	4
16	Kentucky	3
17	Louisiana	5
18	Maine	4
19	Maryland	4
20	Massachusetts[E]	6
21	Michigan	6
22	Minnesota	5
23	Mississippi	4
24	Missouri	6
25	Montana	5
26	Nebraska	6
27	Nevada	3
28	New Hampshire	10

8. List of States and Avg dates for processing (confirmed delivery date – order date) by month.



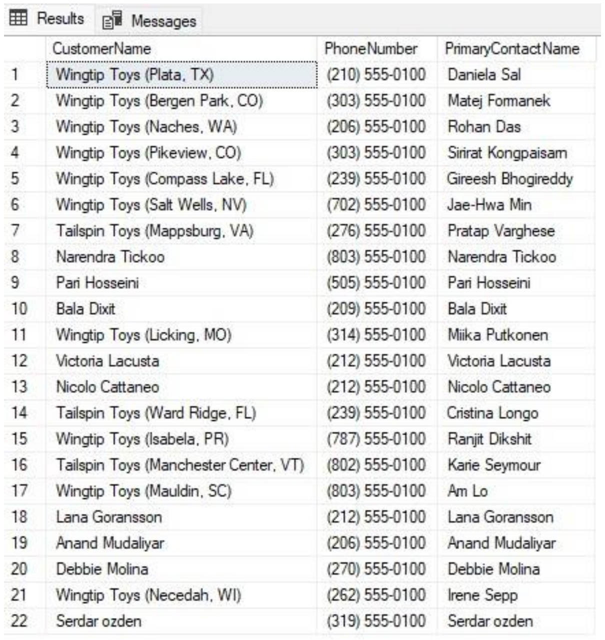
	StateProvinceName	month	avg_processing_dates
1	Alabama	1	10
2	Alabama	2	2
3	Alabama	3	3
4	Alabama	4	4
5	Alabama	5	3
6	Alabama	6	2
7	Alabama	7	11
8	Alabama	8	1
9	Alabama	9	3
10	Alabama	10	5
11	Alabama	11	11
12	Alabama	12	5
13	Alaska	1	1
14	Alaska	2	1
15	Alaska	3	1
16	Alaska	4	2
17	Alaska	5	5
18	Alaska	6	4
19	Alaska	7	4
20	Alaska	8	8
21	Alaska	9	13
22	Alaska	10	5
23	Alaska	11	13
24	Alaska	12	8
25	Arizona	1	2
26	Arizona	2	1
27	Arizona	3	5
28	Arizona	4	5

9. List of StockItems that the company purchased more than sold in the year of 2015.



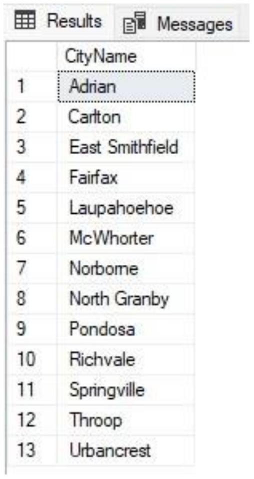
	StockItemName
1	"The Gu" red shirt XML tag t-shirt (White) M
2	Black and orange glass with care despatch tape 48...
3	Tape dispenser (Red)
4	"The Gu" red shirt XML tag t-shirt (White) 5XL
5	Shipping carton (Brown) 305x305x305mm
6	"The Gu" red shirt XML tag t-shirt (Black) 4XL
7	"The Gu" red shirt XML tag t-shirt (White) XS
8	"The Gu" red shirt XML tag t-shirt (Black) XL
9	"The Gu" red shirt XML tag t-shirt (White) XXS

10. List of Customers and their phone number, together with the primary contact person's name, to whom we did not sell more than 10 mugs (search by name) in the year 2016.


Results
Messages

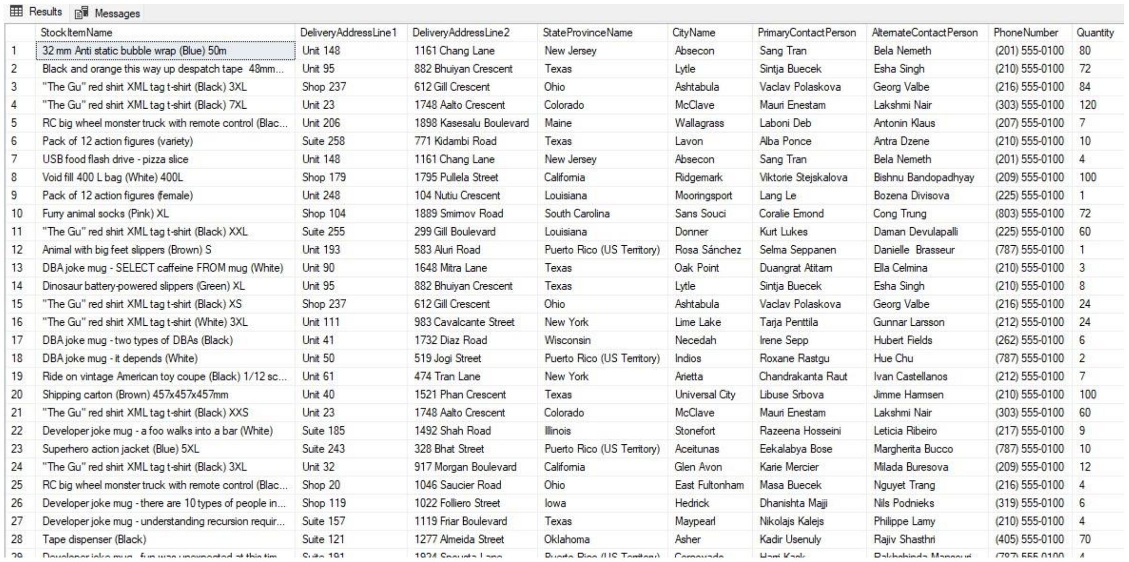
	CustomerName	PhoneNumber	PrimaryContactName
1	Wingtip Toys (Plata, TX)	(210) 555-0100	Daniela Sal
2	Wingtip Toys (Bergen Park, CO)	(303) 555-0100	Matej Formanek
3	Wingtip Toys (Naches, WA)	(206) 555-0100	Rohan Das
4	Wingtip Toys (Pikeview, CO)	(303) 555-0100	Sirirat Kongpaisarn
5	Wingtip Toys (Compass Lake, FL)	(239) 555-0100	Gireesh Bhogireddy
6	Wingtip Toys (Salt Wells, NV)	(702) 555-0100	Jae-Hwa Min
7	Tailspin Toys (Mappsburg, VA)	(276) 555-0100	Pratap Varghese
8	Narendra Tickoo	(803) 555-0100	Narendra Tickoo
9	Pari Hosseini	(505) 555-0100	Pari Hosseini
10	Bala Dixit	(209) 555-0100	Bala Dixit
11	Wingtip Toys (Licking, MO)	(314) 555-0100	Miika Putkonen
12	Victoria Lacusta	(212) 555-0100	Victoria Lacusta
13	Nicolo Cattaneo	(212) 555-0100	Nicolo Cattaneo
14	Tailspin Toys (Ward Ridge, FL)	(239) 555-0100	Cristina Longo
15	Wingtip Toys (Isabela, PR)	(787) 555-0100	Ranjit Dikshit
16	Tailspin Toys (Manchester Center, VT)	(802) 555-0100	Karie Seymour
17	Wingtip Toys (Mauldin, SC)	(803) 555-0100	Am Lo
18	Lana Goransson	(212) 555-0100	Lana Goransson
19	Anand Mudaliyar	(206) 555-0100	Anand Mudaliyar
20	Debbie Molina	(270) 555-0100	Debbie Molina
21	Wingtip Toys (Necedah, WI)	(262) 555-0100	Irene Sepp
22	Serdar ozden	(319) 555-0100	Serdar ozden

11. List all the cities that were updated after 2015-01-01.


Results
Messages

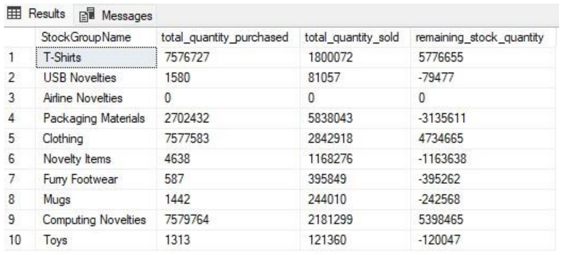
	CityName
1	Adrian
2	Carlton
3	East Smithfield
4	Fairfax
5	Laupahoehoe
6	McWhorter
7	Norborne
8	North Granby
9	Pondosa
10	Richvale
11	Springville
12	Throop
13	Urbancrest

12. List all the Order Detail (Stock Item name, delivery address, delivery state, city, country, customer name, customer contact person name, customer phone, quantity) for the date of 2014-07-01. Info should be relevant to that date.



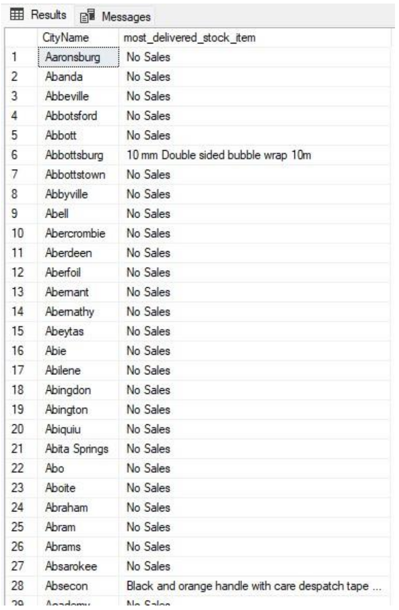
	StockItemName	DeliveryAddressLine1	DeliveryAddressLine2	StateProvinceName	CityName	PrimaryContactPerson	AlternateContactPerson	PhoneNumber	Quantity
1	32 mm Anti static bubble wrap (Blue) 50m	Unit 148	1161 Chang Lane	New Jersey	Absecon	Sang Tran	Bela Nemeth	(201) 555-0100	80
2	Black and orange this way up despatch tape 48mm...	Unit 95	882 Bhuiyan Crescent	Texas	Lytle	Sintja Buecek	Esha Singh	(210) 555-0100	72
3	"The Gu" red shirt XML tag t-shirt (Black) 3XL	Shop 237	612 Gill Crescent	Ohio	Ashtabula	Vaclav Polaskova	Georg Valbe	(216) 555-0100	84
4	"The Gu" red shirt XML tag t-shirt (Black) 7XL	Unit 23	1748 Aalto Crescent	Colorado	McClave	Mauri Enestam	Lakshmi Nair	(303) 555-0100	120
5	RC big wheel monster truck with remote control (Blac...	Unit 206	1898 Kasesalu Boulevard	Maine	Wallagrass	Laboni Deb	Antonin Klaus	(207) 555-0100	7
6	Pack of 12 action figures (variety)	Suite 258	771 Kidambi Road	Texas	Lavon	Alba Ponce	Antra Dzene	(210) 555-0100	10
7	USB food flash drive - pizza slice	Unit 148	1161 Chang Lane	New Jersey	Absecon	Sang Tran	Bela Nemeth	(201) 555-0100	4
8	Void fill 400 L bag (White) 400L	Shop 179	1795 Pullela Street	California	Ridgemark	Viktorie Stejskalova	Bishnu Bandopadhyay	(209) 555-0100	100
9	Pack of 12 action figures (female)	Unit 248	104 Nutiu Crescent	Louisiana	Mooringsport	Lang Le	Bozena Divisova	(225) 555-0100	1
10	Furry animal socks (Pink) XL	Shop 104	1889 Smirnov Road	South Carolina	Sans Souci	Coralie Emond	Cong Trung	(803) 555-0100	72
11	"The Gu" red shirt XML tag t-shirt (Black) XXL	Suite 255	299 Gill Boulevard	Louisiana	Donner	Kurt Lukes	Daman Devulapalli	(225) 555-0100	60
12	Animal with big feet slippers (Brown) S	Unit 193	583 Aluri Road	Puerto Rico (US Territory)	Rosa Sánchez	Selma Seppanen	Danielle Brasseur	(787) 555-0100	1
13	DBA joke mug - SELECT caffeine FROM mug (White)	Unit 90	1648 Mitra Lane	Texas	Oak Point	Duangrat Atitam	Ella Celmina	(210) 555-0100	3
14	Dinosaur battery-powered slippers (Green) XL	Unit 95	882 Bhuiyan Crescent	Texas	Lytle	Sintja Buecek	Esha Singh	(210) 555-0100	8
15	"The Gu" red shirt XML tag t-shirt (Black) XS	Shop 237	612 Gill Crescent	Ohio	Ashtabula	Vaclav Polaskova	Georg Valbe	(216) 555-0100	24
16	"The Gu" red shirt XML tag t-shirt (White) 3XL	Unit 111	983 Cavalcante Street	New York	Lime Lake	Tarja Penttila	Gunnar Larsson	(212) 555-0100	24
17	DBA joke mug - two types of DBAs (Black)	Unit 41	1732 Diaz Road	Wisconsin	Necedah	Irene Sepp	Hubert Fields	(262) 555-0100	6
18	DBA joke mug - it depends (White)	Unit 50	519 Jogi Street	Puerto Rico (US Territory)	Indios	Roxane Rastgu	Hue Chu	(787) 555-0100	2
19	Ride on vintage American toy coupe (Black) 1/12 sc...	Unit 61	474 Tran Lane	New York	Arietta	Chandrakanta Raut	Ivan Castellanos	(212) 555-0100	7
20	Shipping carton (Brown) 457x457x457mm	Unit 40	1521 Phan Crescent	Texas	Universal City	Libuse Srbova	Jimme Hamsen	(210) 555-0100	100
21	"The Gu" red shirt XML tag t-shirt (Black) XXS	Unit 23	1748 Aalto Crescent	Colorado	McClave	Mauri Enestam	Lakshmi Nair	(303) 555-0100	60
22	Developer joke mug - a foo walks into a bar (White)	Suite 185	1492 Shah Road	Illinois	Stonefort	Razeena Hosseini	Leticia Ribeiro	(217) 555-0100	9
23	Superhero action jacket (Blue) 5XL	Suite 243	328 Bhat Street	Puerto Rico (US Territory)	Aceitunas	Eekalabya Bose	Margherita Bucco	(787) 555-0100	10
24	"The Gu" red shirt XML tag t-shirt (Black) 3XL	Unit 32	917 Morgan Boulevard	California	Glen Avon	Karie Mercier	Milada Buresova	(209) 555-0100	12
25	RC big wheel monster truck with remote control (Blac...	Shop 20	1046 Saucier Road	Ohio	East Fultonham	Masa Buecek	Nguyet Trang	(216) 555-0100	4
26	Developer joke mug - there are 10 types of people in...	Shop 119	1022 Folliero Street	Iowa	Hedrick	Dhanishta Majji	Nils Podnieks	(319) 555-0100	6
27	Developer joke mug - understanding recursion requir...	Suite 157	1119 Friar Boulevard	Texas	Maypearl	Nikolajs Kalejs	Philippe Lamy	(210) 555-0100	4
28	Tape dispenser (Black)	Suite 121	1277 Almeida Street	Oklahoma	Asher	Kadir Usenuly	Rajiv Shasthri	(405) 555-0100	70

13. List of stock item groups and total quantity purchased, total quantity sold, and the remaining stock quantity (quantity purchased – quantity sold)



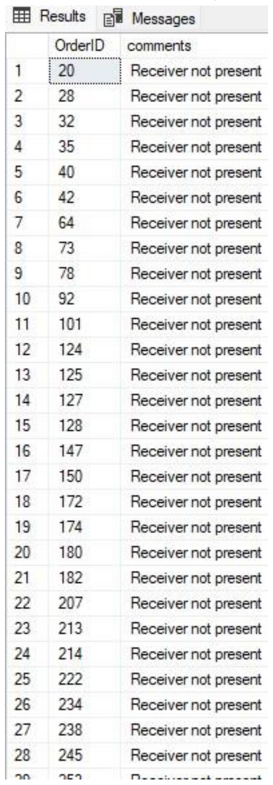
	StockGroupName	total_quantity_purchased	total_quantity_sold	remaining_stock_quantity
1	T-Shirts	7576727	1800072	5776655
2	USB Novelties	1580	81057	-79477
3	Airline Novelties	0	0	0
4	Packaging Materials	2702432	5838043	-3135611
5	Clothing	7577583	2842918	4734665
6	Novelty Items	4638	1168276	-1163638
7	Furry Footwear	587	395849	-395262
8	Mugs	1442	244010	-242568
9	Computing Novelties	7579764	2181299	5398465
10	Toys	1313	121360	-120047

14. List of Cities in the US and the stock item that the city got the most deliveries in 2016. If the city did not purchase any stock items in 2016, print "No Sales".



	CityName	most_delivered_stock_item
1	Aaronsburg	No Sales
2	Abanda	No Sales
3	Abbeville	No Sales
4	Abbotsford	No Sales
5	Abbott	No Sales
6	Abbottsburg	10 mm Double sided bubble wrap 10m
7	Abbottstown	No Sales
8	Abbyville	No Sales
9	Abell	No Sales
10	Abercrombie	No Sales
11	Aberdeen	No Sales
12	Aberfoil	No Sales
13	Abernant	No Sales
14	Abernathy	No Sales
15	Abeytas	No Sales
16	Abie	No Sales
17	Abilene	No Sales
18	Abingdon	No Sales
19	Abington	No Sales
20	Abiquiu	No Sales
21	Abita Springs	No Sales
22	Abo	No Sales
23	Aboite	No Sales
24	Abraham	No Sales
25	Abram	No Sales
26	Abrams	No Sales
27	Absarokee	No Sales
28	Absecon	Black and orange handle with care despatch tape ...

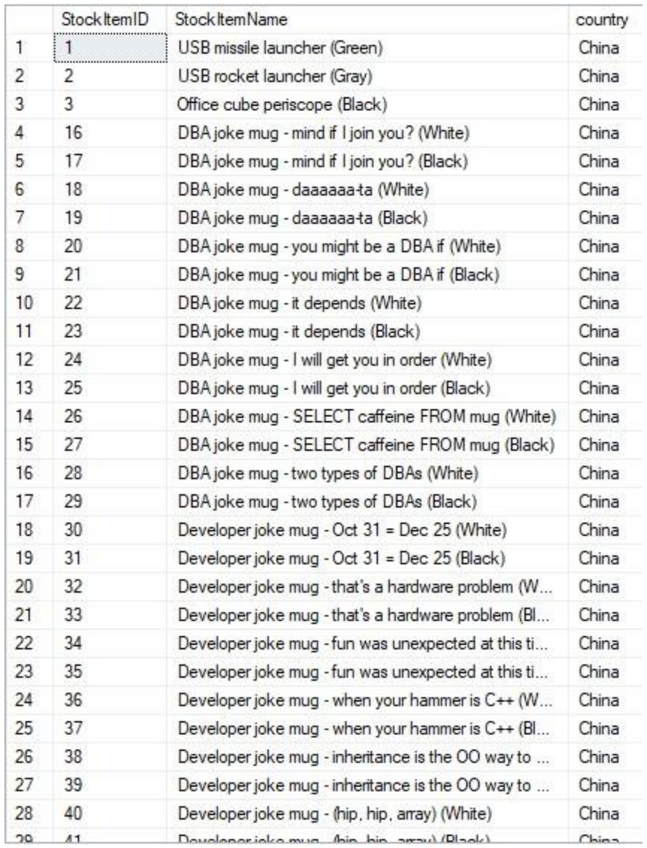
15. List any orders that had more than one delivery attempt (located in invoice table).



Results | Messages

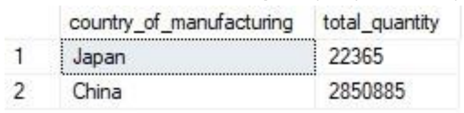
	OrderID	comments
1	20	Receiver not present
2	28	Receiver not present
3	32	Receiver not present
4	35	Receiver not present
5	40	Receiver not present
6	42	Receiver not present
7	64	Receiver not present
8	73	Receiver not present
9	78	Receiver not present
10	92	Receiver not present
11	101	Receiver not present
12	124	Receiver not present
13	125	Receiver not present
14	127	Receiver not present
15	128	Receiver not present
16	147	Receiver not present
17	150	Receiver not present
18	172	Receiver not present
19	174	Receiver not present
20	180	Receiver not present
21	182	Receiver not present
22	207	Receiver not present
23	213	Receiver not present
24	214	Receiver not present
25	222	Receiver not present
26	234	Receiver not present
27	238	Receiver not present
28	245	Receiver not present

16. List all stock items that are manufactured in China. (Country of Manufacture)



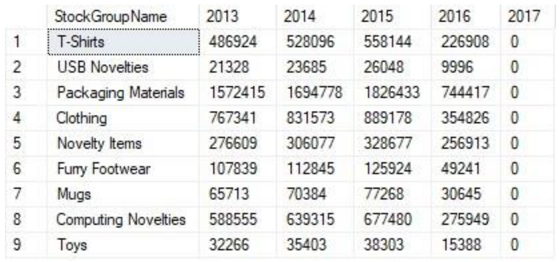
	StockItemID	StockItemName	country
1	1	USB missile launcher (Green)	China
2	2	USB rocket launcher (Gray)	China
3	3	Office cube periscope (Black)	China
4	16	DBA joke mug - mind if I join you? (White)	China
5	17	DBA joke mug - mind if I join you? (Black)	China
6	18	DBA joke mug - daaaaaa-ta (White)	China
7	19	DBA joke mug - daaaaaa-ta (Black)	China
8	20	DBA joke mug - you might be a DBA if (White)	China
9	21	DBA joke mug - you might be a DBA if (Black)	China
10	22	DBA joke mug - it depends (White)	China
11	23	DBA joke mug - it depends (Black)	China
12	24	DBA joke mug - I will get you in order (White)	China
13	25	DBA joke mug - I will get you in order (Black)	China
14	26	DBA joke mug - SELECT caffeine FROM mug (White)	China
15	27	DBA joke mug - SELECT caffeine FROM mug (Black)	China
16	28	DBA joke mug - two types of DBAs (White)	China
17	29	DBA joke mug - two types of DBAs (Black)	China
18	30	Developer joke mug - Oct 31 = Dec 25 (White)	China
19	31	Developer joke mug - Oct 31 = Dec 25 (Black)	China
20	32	Developer joke mug - that's a hardware problem (W...	China
21	33	Developer joke mug - that's a hardware problem (Bl...	China
22	34	Developer joke mug - fun was unexpected at this ti...	China
23	35	Developer joke mug - fun was unexpected at this ti...	China
24	36	Developer joke mug - when your hammer is C++ (W...	China
25	37	Developer joke mug - when your hammer is C++ (Bl...	China
26	38	Developer joke mug - inheritance is the OO way to ...	China
27	39	Developer joke mug - inheritance is the OO way to ...	China
28	40	Developer joke mug - (hip, hip, array) (White)	China

17. Total quantity of stock items sold in 2015, group by country of manufacturing.



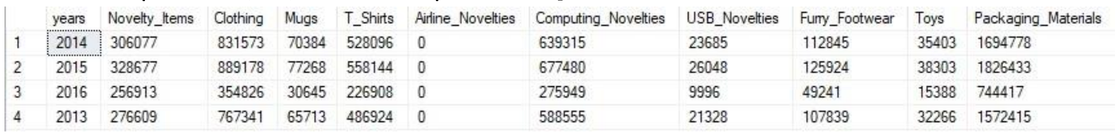
	country_of_manufacturing	total_quantity
1	Japan	22365
2	China	2850885

18. Create a view that shows the total quantity of stock items of each stock group sold (in orders) by year 2013-2017. [Stock Group Name, 2013, 2014, 2015, 2016, 2017]



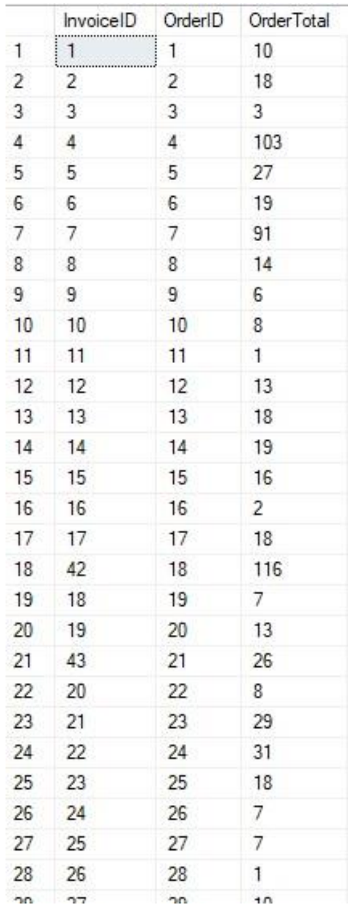
	StockGroupName	2013	2014	2015	2016	2017
1	T-Shirts	486924	528096	558144	226908	0
2	USB Novelties	21328	23685	26048	9996	0
3	Packaging Materials	1572415	1694778	1826433	744417	0
4	Clothing	767341	831573	889178	354826	0
5	Novelty Items	276609	306077	328677	256913	0
6	Furry Footwear	107839	112845	125924	49241	0
7	Mugs	65713	70384	77268	30645	0
8	Computing Novelties	588555	639315	677480	275949	0
9	Toys	32266	35403	38303	15388	0

19. Create a view that shows the total quantity of stock items of each stock group sold (in orders) by year 2013-2017. [Year, Stock Group Name1, Stock Group Name2, Stock Group Name3, … , Stock Group Name10]



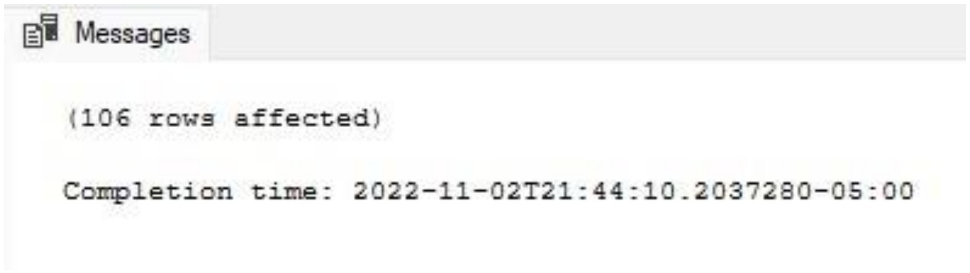
	years	Novelty_Items	Clothing	Mugs	T_Shirts	Airline_Novelties	Computing_Novelties	USB_Novelties	Furry_Footwear	Toys	Packaging_Materials
1	2014	306077	831573	70384	528096	0	639315	23685	112845	35403	1694778
2	2015	328677	889178	77268	558144	0	677480	26048	125924	38303	1826433
3	2016	256913	354826	30645	226908	0	275949	9996	49241	15388	744417
4	2013	276609	767341	65713	486924	0	588555	21328	107839	32266	1572415

20. Create a function, input: order id; return: total of that order. List invoices and use that function to attach the order total to the other fields of invoices.



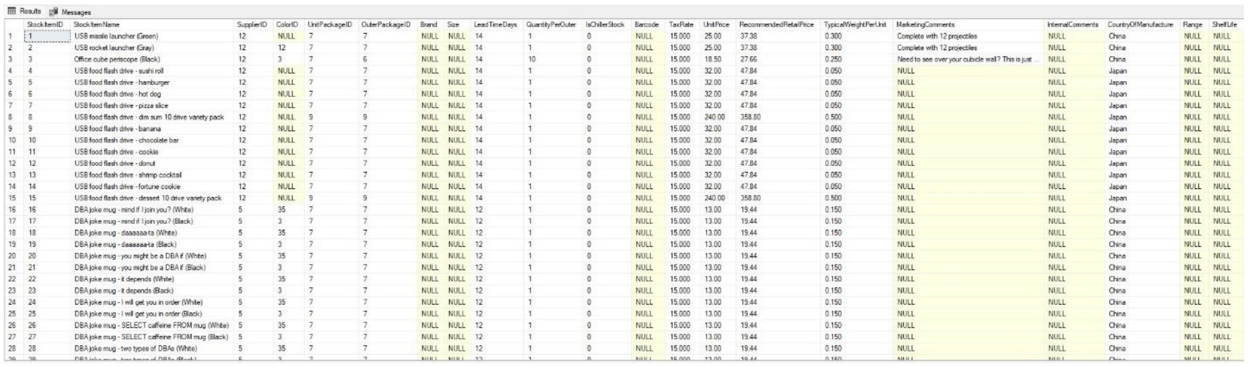
	InvoiceID	OrderID	OrderTotal
1	1	1	10
2	2	2	18
3	3	3	3
4	4	4	103
5	5	5	27
6	6	6	19
7	7	7	91
8	8	8	14
9	9	9	6
10	10	10	8
11	11	11	1
12	12	12	13
13	13	13	18
14	14	14	19
15	15	15	16
16	16	16	2
17	17	17	18
18	42	18	116
19	18	19	7
20	19	20	13
21	43	21	26
22	20	22	8
23	21	23	29
24	22	24	31
25	23	25	18
26	24	26	7
27	25	27	7
28	26	28	1

21. Create a new table called ods.Orders. Create a stored procedure, with proper error handling and transactions, that input is a date; when executed, it would find orders of that day, calculate order total, and save the information (order id, order date, order total, customer id) into the new table. If a given date is already existing in the new table, throw an error and roll back. Execute the stored procedure 5 times using different dates.



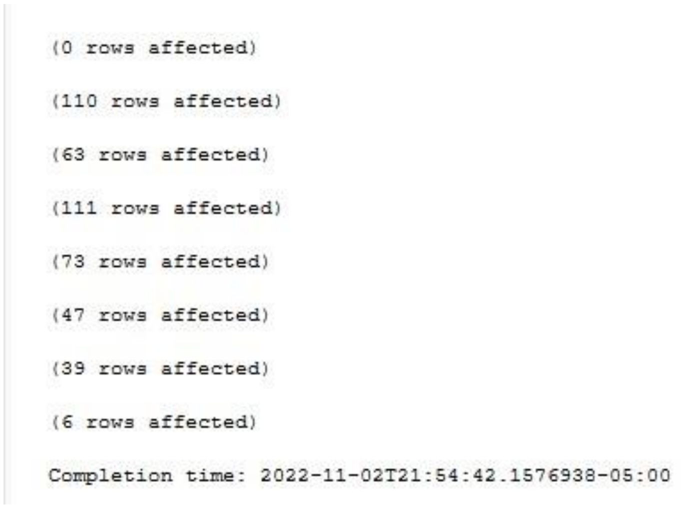
```
(106 rows affected)

Completion time: 2022-11-02T21:44:10.2037280-05:00
```

22. Create a new table called ods.StockItem. It has following columns: [StockItemID], [StockItemName] ,[SupplierID] ,[ColorID] ,[UnitPackageID] ,[OuterPackageID] ,[Brand] ,[Size] ,[LeadTimeDays] ,[QuantityPerOuter] ,[IsChillerStock] ,[Barcode] ,[TaxRate] ,[UnitPrice],[RecommendedRetailPrice] ,[TypicalWeightPerUnit] ,[MarketingComments] ,[InternalComments], [CountryOfManufacture], [Range], [Shelflife]. Migrate all the data in the original stock item table.



	StockItemID	StockItemName	SupplierID	ColorID	UnitPackageID	OuterPackageID	Brand	Size	LeadTimeDays	QuantityPerOuter	IsChillerStock	Barcode	TaxRate	UnitPrice	RecommendedRetailPrice	TypicalWeightPerUnit	MarketingComments	InternalComments	CountryOfManufacture	Range	ShelfLife
1	1	USB missile launcher (Green)	12	NULL	7	7	NULL	NULL	14	1	0	NULL	15.000	25.00	37.38	0.300	Complete with 12 projectiles	NULL	China	NULL	NULL
2	2	USB rocket launcher (Gray)	12	12	7	7	NULL	NULL	14	1	0	NULL	15.000	25.00	37.38	0.300	Complete with 12 projectiles	NULL	China	NULL	NULL
3	3	Office cube periscope (Black)	12	3	7	6	NULL	NULL	14	10	0	NULL	15.000	18.50	27.66	0.250	Need to see over your cubicle wall? This is just ...	NULL	China	NULL	NULL
4	4	USB food flash drive - sushi roll	12	NULL	7	7	NULL	NULL	14	1	0	NULL	15.000	32.00	47.84	0.050	NULL	NULL	Japan	NULL	NULL
5	5	USB food flash drive - hamburger	12	NULL	7	7	NULL	NULL	14	1	0	NULL	15.000	32.00	47.84	0.050	NULL	NULL	Japan	NULL	NULL
6	6	USB food flash drive - hot dog	12	NULL	7	7	NULL	NULL	14	1	0	NULL	15.000	32.00	47.84	0.050	NULL	NULL	Japan	NULL	NULL
7	7	USB food flash drive - pizza slice	12	NULL	7	7	NULL	NULL	14	1	0	NULL	15.000	32.00	47.84	0.050	NULL	NULL	Japan	NULL	NULL
8	8	USB food flash drive - dim sum 10 drive variety pack	12	NULL	9	9	NULL	NULL	14	1	0	NULL	15.000	240.00	358.80	0.500	NULL	NULL	Japan	NULL	NULL
9	9	USB food flash drive - banana	12	NULL	7	7	NULL	NULL	14	1	0	NULL	15.000	32.00	47.84	0.050	NULL	NULL	Japan	NULL	NULL
10	10	USB food flash drive - chocolate bar	12	NULL	7	7	NULL	NULL	14	1	0	NULL	15.000	32.00	47.84	0.050	NULL	NULL	Japan	NULL	NULL
11	11	USB food flash drive - cookie	12	NULL	7	7	NULL	NULL	14	1	0	NULL	15.000	32.00	47.84	0.050	NULL	NULL	Japan	NULL	NULL
12	12	USB food flash drive - donut	12	NULL	7	7	NULL	NULL	14	1	0	NULL	15.000	32.00	47.84	0.050	NULL	NULL	Japan	NULL	NULL
13	13	USB food flash drive - shrimp cocktail	12	NULL	7	7	NULL	NULL	14	1	0	NULL	15.000	32.00	47.84	0.050	NULL	NULL	Japan	NULL	NULL
14	14	USB food flash drive - fortune cookie	12	NULL	7	7	NULL	NULL	14	1	0	NULL	15.000	32.00	47.84	0.050	NULL	NULL	Japan	NULL	NULL
15	15	USB food flash drive - dessert 10 drive variety pack	12	NULL	9	9	NULL	NULL	14	1	0	NULL	15.000	240.00	358.80	0.500	NULL	NULL	Japan	NULL	NULL
16	16	DBA joke mug - mind if I join you? (White)	5	35	7	7	NULL	NULL	12	1	0	NULL	15.000	13.00	19.44	0.150	NULL	NULL	China	NULL	NULL
17	17	DBA joke mug - mind if I join you? (Black)	5	3	7	7	NULL	NULL	12	1	0	NULL	15.000	13.00	19.44	0.150	NULL	NULL	China	NULL	NULL
18	18	DBA joke mug - daaaaaa-ta (White)	5	35	7	7	NULL	NULL	12	1	0	NULL	15.000	13.00	19.44	0.150	NULL	NULL	China	NULL	NULL
19	19	DBA joke mug - daaaaaa-ta (Black)	5	3	7	7	NULL	NULL	12	1	0	NULL	15.000	13.00	19.44	0.150	NULL	NULL	China	NULL	NULL
20	20	DBA joke mug - you might be a DBA if (White)	5	35	7	7	NULL	NULL	12	1	0	NULL	15.000	13.00	19.44	0.150	NULL	NULL	China	NULL	NULL
21	21	DBA joke mug - you might be a DBA if (Black)	5	3	7	7	NULL	NULL	12	1	0	NULL	15.000	13.00	19.44	0.150	NULL	NULL	China	NULL	NULL
22	22	DBA joke mug - it depends (White)	5	35	7	7	NULL	NULL	12	1	0	NULL	15.000	13.00	19.44	0.150	NULL	NULL	China	NULL	NULL
23	23	DBA joke mug - it depends (Black)	5	3	7	7	NULL	NULL	12	1	0	NULL	15.000	13.00	19.44	0.150	NULL	NULL	China	NULL	NULL
24	24	DBA joke mug - I will get you in order (White)	5	35	7	7	NULL	NULL	12	1	0	NULL	15.000	13.00	19.44	0.150	NULL	NULL	China	NULL	NULL
25	25	DBA joke mug - I will get you in order (Black)	5	3	7	7	NULL	NULL	12	1	0	NULL	15.000	13.00	19.44	0.150	NULL	NULL	China	NULL	NULL
26	26	DBA joke mug - SELECT caffeine FROM mug (White)	5	35	7	7	NULL	NULL	12	1	0	NULL	15.000	13.00	19.44	0.150	NULL	NULL	China	NULL	NULL
27	27	DBA joke mug - SELECT caffeine FROM mug (Black)	5	3	7	7	NULL	NULL	12	1	0	NULL	15.000	13.00	19.44	0.150	NULL	NULL	China	NULL	NULL
28	28	DBA joke mug - two types of DBAs (White)	5	35	7	7	NULL	NULL	12	1	0	NULL	15.000	13.00	19.44	0.150	NULL	NULL	China	NULL	NULL

23. Rewrite your stored procedure in (21). Now with a given date, it should wipe out all the order data prior to the input date and load the order data that was placed in the next 7 days following the input date.



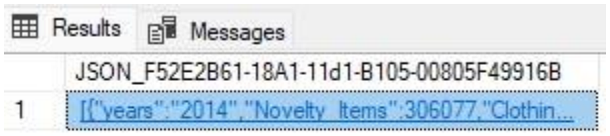
```
(0 rows affected)

(110 rows affected)

(63 rows affected)

(111 rows affected)

(73 rows affected)

(47 rows affected)

(39 rows affected)

(6 rows affected)

Completion time: 2022-11-02T21:54:42.1576938-05:00
```

24. Consider the JSON file:

```
{
  "PurchaseOrders":[
    {
      "StockItemName":"Panzer Video Game",
      "Supplier":"7",
      "UnitPackageId":"1",
      "OuterPackageId":[
        6,
        7
      ],
      "Brand":"EA Sports",
      "LeadTimeDays":"5",
      "QuantityPerOuter":"1",
      "TaxRate":"6",
      "UnitPrice":"59.99",
      "RecommendedRetailPrice":"69.99",
      "TypicalWeightPerUnit":"0.5",
      "CountryOfManufacture":"Canada",
      "Range":"Adult",
      "OrderDate":"2018-01-01",
      "DeliveryMethod":"Post",
      "ExpectedDeliveryDate":"2018-02-02",
      "SupplierReference":"WWI2308"
    },
    {
      "StockItemName":"Panzer Video Game",
      "Supplier":"5",
      "UnitPackageId":"1",
      "OuterPackageId":"7",
      "Brand":"EA Sports",
      "LeadTimeDays":"5",
      "QuantityPerOuter":"1",
      "TaxRate":"6",
      "UnitPrice":"59.99",
      "RecommendedRetailPrice":"69.99",
      "TypicalWeightPerUnit":"0.5",
      "CountryOfManufacture":"Canada",
      "Range":"Adult",
      "OrderDate":"2018-01-025",
      "DeliveryMethod":"Post",
      "ExpectedDeliveryDate":"2018-02-02",
      "SupplierReference":"269622390"
    }
```

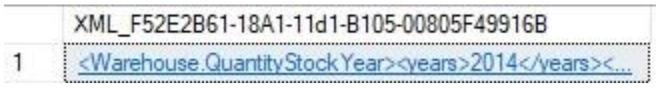
    ]
}
Looks like that it is our missed purchase orders. Migrate these data into Stock Item, Purchase Order and Purchase Order Lines tables. Of course, save the script.

25. Revisit your answer in (19). Convert the result in JSON string and save it to the server using TSQL FOR JSON PATH.



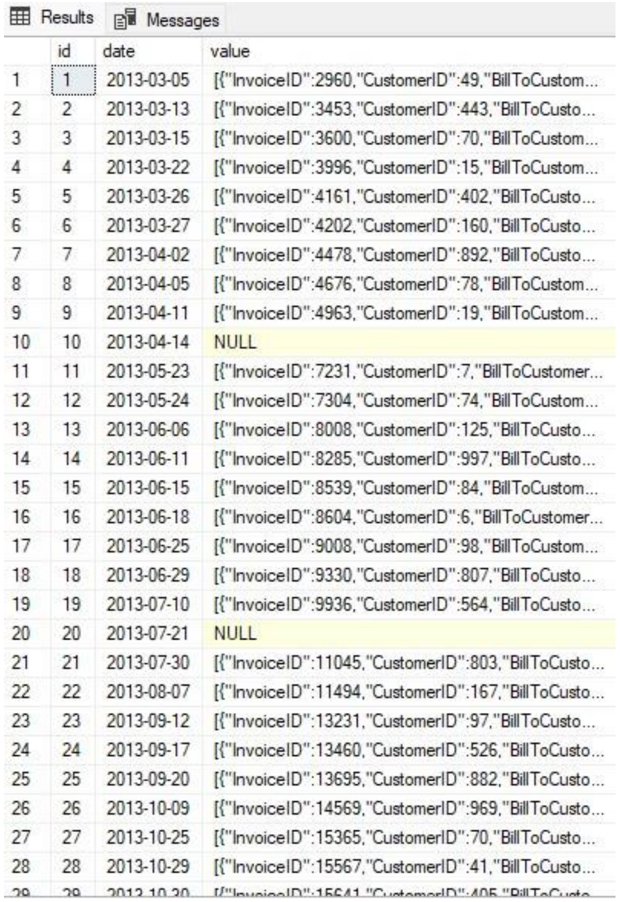
	JSON_F52E2B61-18A1-11d1-B105-00805F49916B
1	[{"years":"2014","Novelty_Items":306077,"Clothin...

26. Revisit your answer in (19). Convert the result into an XML string and save it to the server using TSQL FOR XML PATH.



	XML_F52E2B61-18A1-11d1-B105-00805F49916B
1	<Warehouse.QuantityStockYear><years>2014</years><...

27. Create a new table called ods.ConfirmedDeviveryJson with 3 columns (id, date, value) . Create a stored procedure, input is a date. The logic would load invoice information (all columns) as well as invoice line information (all columns) and forge them into a JSON string and then insert into the new table just created. Then write a query to run the stored procedure for each DATE that customer id 1 got something delivered to him.



	id	date	value
1	1	2013-03-05	[{"InvoiceID":2960,"CustomerID":49,"BillToCustom...
2	2	2013-03-13	[{"InvoiceID":3453,"CustomerID":443,"BillToCusto...
3	3	2013-03-15	[{"InvoiceID":3600,"CustomerID":70,"BillToCustom...
4	4	2013-03-22	[{"InvoiceID":3996,"CustomerID":15,"BillToCustom...
5	5	2013-03-26	[{"InvoiceID":4161,"CustomerID":402,"BillToCusto...
6	6	2013-03-27	[{"InvoiceID":4202,"CustomerID":160,"BillToCusto...
7	7	2013-04-02	[{"InvoiceID":4478,"CustomerID":892,"BillToCusto...
8	8	2013-04-05	[{"InvoiceID":4676,"CustomerID":78,"BillToCustom...
9	9	2013-04-11	[{"InvoiceID":4963,"CustomerID":19,"BillToCustom...
10	10	2013-04-14	NULL
11	11	2013-05-23	[{"InvoiceID":7231,"CustomerID":7,"BillToCustomer...
12	12	2013-05-24	[{"InvoiceID":7304,"CustomerID":74,"BillToCustom...
13	13	2013-06-06	[{"InvoiceID":8008,"CustomerID":125,"BillToCusto...
14	14	2013-06-11	[{"InvoiceID":8285,"CustomerID":997,"BillToCusto...
15	15	2013-06-15	[{"InvoiceID":8539,"CustomerID":84,"BillToCustom...
16	16	2013-06-18	[{"InvoiceID":8604,"CustomerID":6,"BillToCustomer...
17	17	2013-06-25	[{"InvoiceID":9008,"CustomerID":98,"BillToCustom...
18	18	2013-06-29	[{"InvoiceID":9330,"CustomerID":807,"BillToCusto...
19	19	2013-07-10	[{"InvoiceID":9936,"CustomerID":564,"BillToCusto...
20	20	2013-07-21	NULL
21	21	2013-07-30	[{"InvoiceID":11045,"CustomerID":803,"BillToCusto...
22	22	2013-08-07	[{"InvoiceID":11494,"CustomerID":167,"BillToCusto...
23	23	2013-09-12	[{"InvoiceID":13231,"CustomerID":97,"BillToCusto...
24	24	2013-09-17	[{"InvoiceID":13460,"CustomerID":526,"BillToCusto...
25	25	2013-09-20	[{"InvoiceID":13695,"CustomerID":882,"BillToCusto...
26	26	2013-10-09	[{"InvoiceID":14569,"CustomerID":969,"BillToCusto...
27	27	2013-10-25	[{"InvoiceID":15365,"CustomerID":70,"BillToCusto...
28	28	2013-10-29	[{"InvoiceID":15567,"CustomerID":41,"BillToCusto...
29	29	2013-10-30	[{"InvoiceID":15641,"CustomerID":405,"BillToCusto

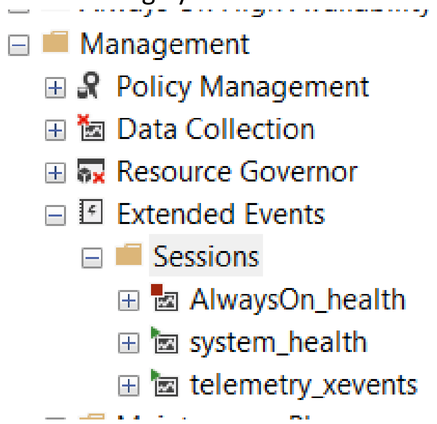
28. Write a short essay talking about your understanding of transactions, locks, and isolation levels.

    Transaction is a logical work unit that performs one or more activities. It is also SQL's save/undo button. It meets ACID principle, which stands for atomicity, consistency, isolation, and durability. In addition, transaction is a set of commands that must be executed as a set. If one of the commands in the transaction cannot be executed, the whole transaction will not be executed. Besides, transaction has two outcomes which are committed or rolled back. There are multiple types of transactions including autocommit transaction, implicit transaction, explicit transaction, and batch-scoped transaction.

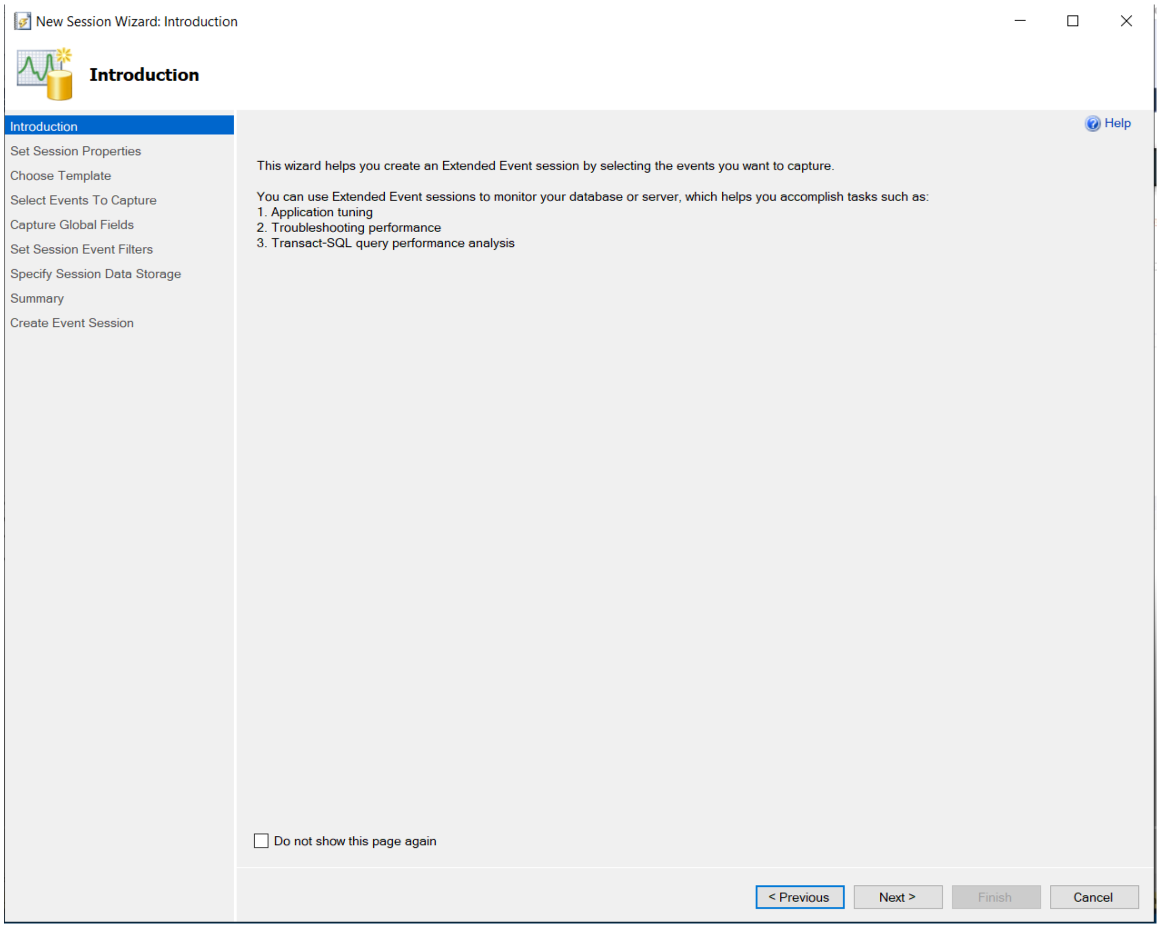
    Locks are used in Pessimistic Concurrency Control to prevent users from modifying data in a way that affects other users. If a lock is applied, other users need to wait until the transaction is committed/rollbacked and then can start their own transactions. Lock types include exclusive lock (will ensure that a page or row will be reserved exclusively for the transaction that imposed the exclusive lock, as long as the transaction holds the lock), shared lock (will reserve a page or row to be available only for reading ○ can be imposed by several transactions at the same time, will allow write operations, but no DDL changes will be allowed), update lock(can be imposed on a record that already has a shared lock. In such a case, the update lock will impose another shared lock on the target row. Once the transaction that holds the update lock is ready to change the data, the update lock (U) will be transformed to an exclusive lock (X)), intent lock (used by a transaction to inform another transaction about its intention to acquire a lock), schema lock, and bulk update lock (designed to be used by bulk import operations).

    There are five isolation levels, all of which are used to guarantee the accuracy of the data (i.e., make sure the data is the most up-to-date version). The five levels are: Read Uncommitted, Read Committed, Repeatable Read, Serializable, and Snapshot. Read Uncommitted is the first level of isolation, and it comes under the pessimistic model of concurrency. In Read Uncommitted, one transaction is allowed to read the data that is about to be changed by the commit of another process. Read Uncommitted allows the dirty read problem. Read Committed is the system default. This is the second level of isolation and also falls under the pessimistic model of concurrency. In the Read Committed, we are only allowed to read data that is committed, which means this level eliminates the dirty read problem. Repeatable Read is similar to the Read Committed level and eliminates the NonRepeatable Read problem. In this level, the transaction has to wait till another transaction's update or read query is complete. But if there is an insert transaction, it does not wait for anyone. This can lead to the Phantom Read problem. Serializable is the highest level of isolation in the pessimistic model. ○ In this level of isolation, we can ask any transaction to wait until the current transaction completes. By implementing this level of isolation, we can prevent the Phantom Read problem. Snapshot follows the optimistic model of concurrency. It avoids most locking and blocking by using row versioning. When data is modified, the committed versions of affected rows are copied to tempdb and given version numbers. This operation is called copy on write and is used for all inserts, updates and deletes using this technique. When another session reads the same data, the committed version of the data as of the time the reading transaction began is returned.

29. Write a short essay, plus screenshots talking about performance tuning in SQL Server. Must include Tuning Advisor, Extended Events, DMV, Logs and Execution Plan.
    1) Extended Events:
       Using SQL Server Event Bubbling System



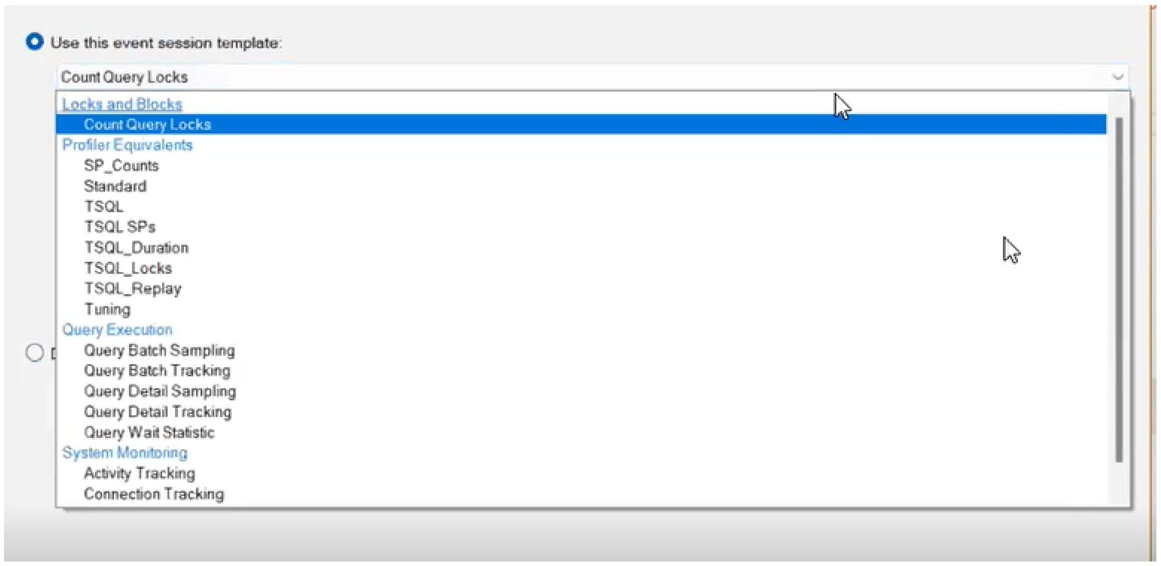
Start a new session using New Session Wizard:



This wizard helps you create an Extended Event session by selecting the events you want to capture.

You can use Extended Event sessions to monitor your database or server, which helps you accomplish tasks such as:
1. Application tuning
2. Troubleshooting performance
3. Transact-SQL query performance analysis

Choose a template:

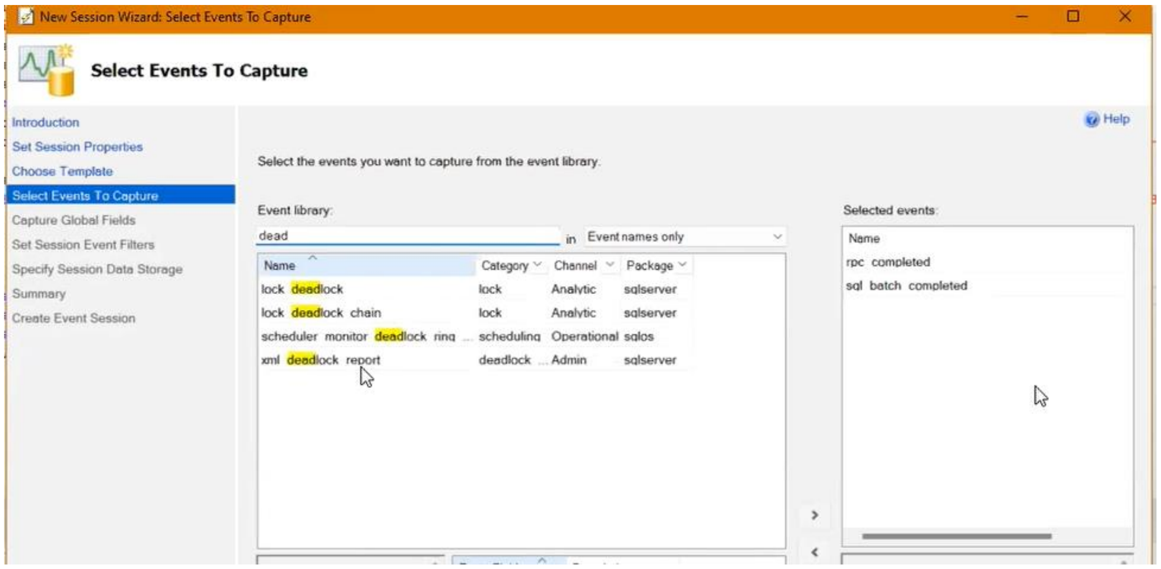


Use this event session template:

Count Query Locks

- Locks and Blocks
  - Count Query Locks
- Profiler Equivalents
  - SP_Counts
  - Standard
  - TSQL
  - TSQL SPs
  - TSQL_Duration
  - TSQL_Locks
  - TSQL_Replay
  - Tuning
- Query Execution
  - Query Batch Sampling
  - Query Batch Tracking
  - Query Detail Sampling
  - Query Detail Tracking
  - Query Wait Statistic
- System Monitoring
  - Activity Tracking
  - Connection Tracking

Select Events to capture, the right column are events that are already selected:



Select Events To Capture

Select the events you want to capture from the event library.

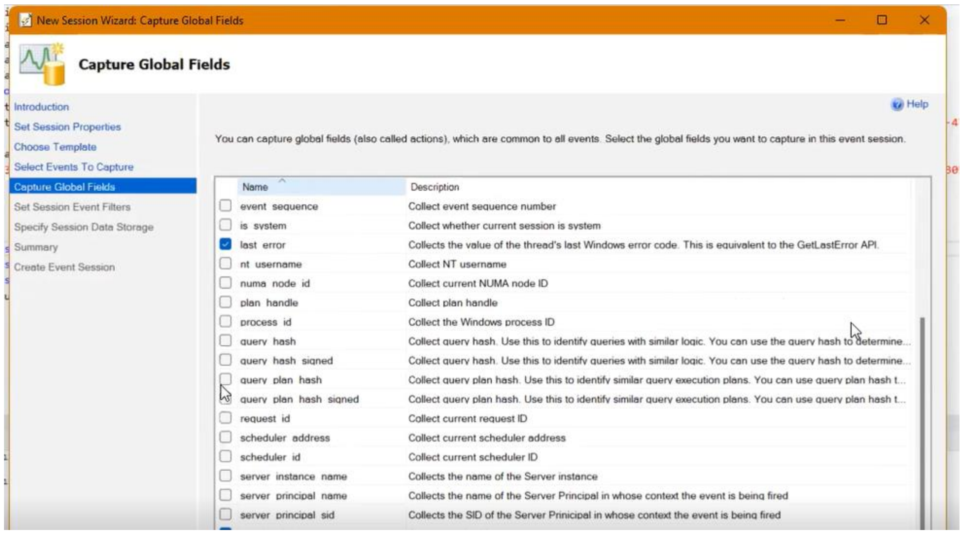
Event library: dead in Event names only

Name	Category	Channel	Package
lock_deadlock	lock	Analytic	sqlserver
lock_deadlock_chain	lock	Analytic	sqlserver
scheduler_monitor_deadlock_ring ...	scheduling	Operational	sqlos
xml_deadlock_report	deadlock ...	Admin	sqlserver

Selected events:

Name
rpc_completed
sql_batch_completed

Choose data other than events that you want to capture:

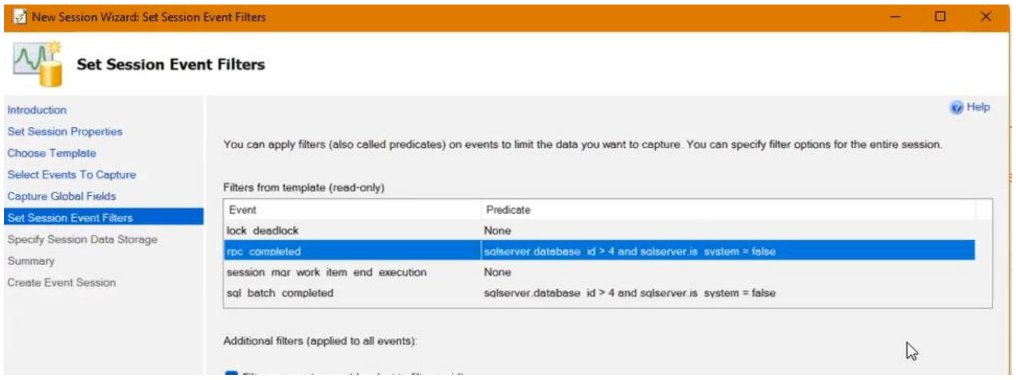


Capture Global Fields

You can capture global fields (also called actions), which are common to all events. Select the global fields you want to capture in this event session.

	Name	Description
☐	event_sequence	Collect event sequence number
☐	is_system	Collect whether current session is system
✓	last_error	Collects the value of the thread's last Windows error code. This is equivalent to the GetLastError API.
☐	nt_username	Collect NT username
☐	numa_node_id	Collect current NUMA node ID
☐	plan_handle	Collect plan handle
☐	process_id	Collect the Windows process ID
☐	query_hash	Collect query hash. Use this to identify queries with similar logic. You can use the query hash to determine...
☐	query_hash_signed	Collect query hash. Use this to identify queries with similar logic. You can use the query hash to determine...
☐	query_plan_hash	Collect query plan hash. Use this to identify similar query execution plans. You can use query plan hash t...
☐	query_plan_hash_signed	Collect query plan hash. Use this to identify similar query execution plans. You can use query plan hash t...
☐	request_id	Collect current request ID
☐	scheduler_address	Collect current scheduler address
☐	scheduler_id	Collect current scheduler ID
☐	server_instance_name	Collects the name of the Server instance
☐	server_principal_name	Collects the name of the Server Principal in whose context the event is being fired
☐	server_principal_sid	Collects the SID of the Server Prinicipal in whose context the event is being fired

Specify filter so that can customize threshold (e.g., specify what should be the maximum time before the query is captured as long running query.



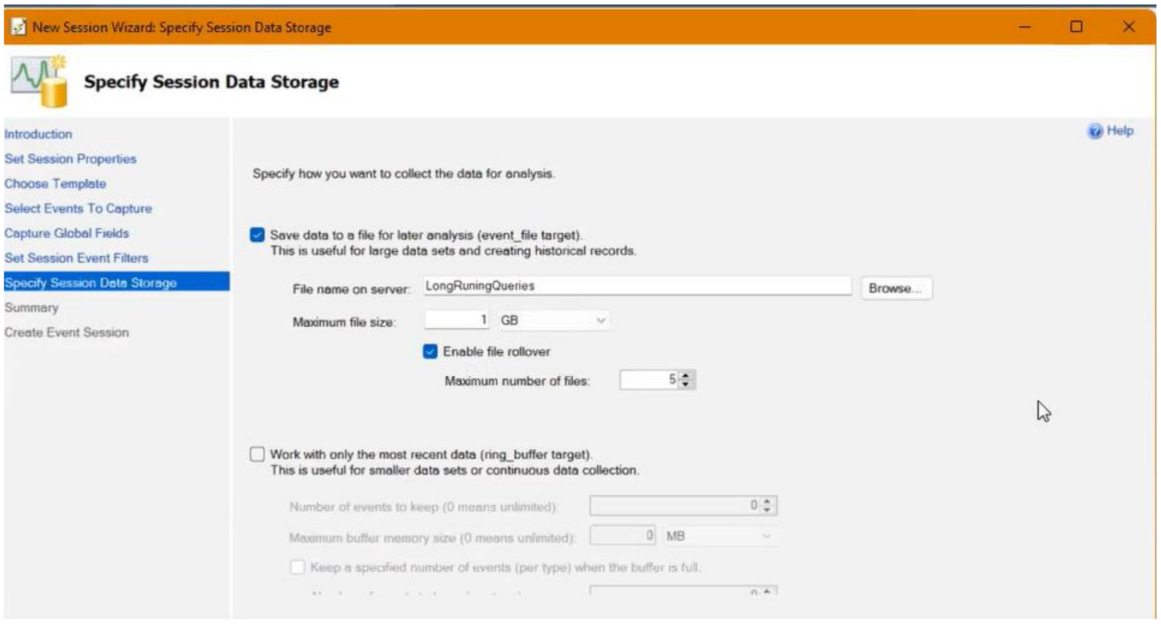
You can apply filters (also called predicates) on events to limit the data you want to capture. You can specify filter options for the entire session.

Filters from template (read-only)

Event	Predicate
lock_deadlock	None
rpc_completed	sqlserver.database_id > 4 and sqlserver.is_system = false
session_mgr_work_item_end_execution	None
sql_batch_completed	sqlserver.database_id > 4 and sqlserver.is_system = false

Additional filters (applied to all events):

Choose if want to store the data for later analysis:



Specify how you want to collect the data for analysis.

[✓] Save data to a file for later analysis (event_file target).
This is useful for large data sets and creating historical records.

File name on server: LongRuningQueries

Maximum file size: 1 GB

[✓] Enable file rollover

Maximum number of files: 5

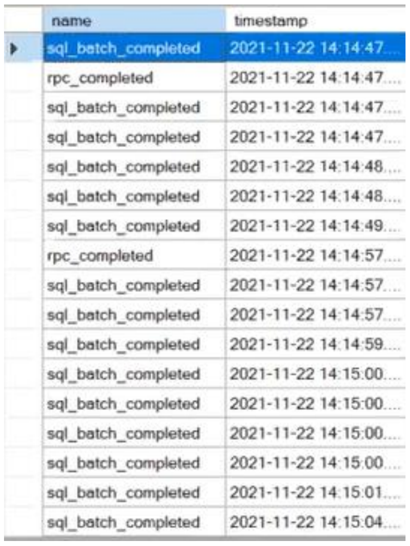
[ ] Work with only the most recent data (ring_buffer target).
This is useful for smaller data sets or continuous data collection.

Number of events to keep (0 means unlimited): 0

Maximum buffer memory size (0 means unlimited): 0 MB

[ ] Keep a specified number of events (per type) when the buffer is full.

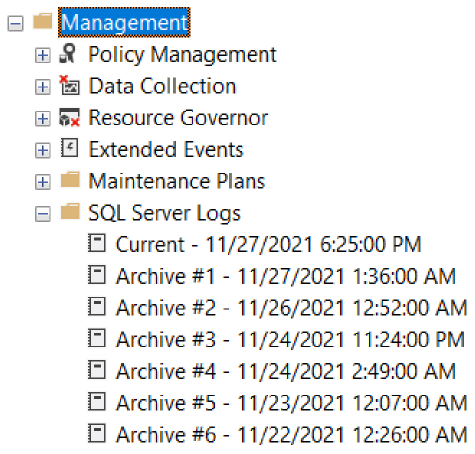
After setting up and running some queries, there will be logs that captures all the actions:



name	timestamp
sql_batch_completed	2021-11-22 14:14:47....
rpc_completed	2021-11-22 14:14:47....
sql_batch_completed	2021-11-22 14:14:47....
sql_batch_completed	2021-11-22 14:14:47....
sql_batch_completed	2021-11-22 14:14:48....
sql_batch_completed	2021-11-22 14:14:48....
sql_batch_completed	2021-11-22 14:14:49....
rpc_completed	2021-11-22 14:14:57....
sql_batch_completed	2021-11-22 14:14:57....
sql_batch_completed	2021-11-22 14:14:57....
sql_batch_completed	2021-11-22 14:14:59....
sql_batch_completed	2021-11-22 14:15:00....
sql_batch_completed	2021-11-22 14:15:00....
sql_batch_completed	2021-11-22 14:15:00....
sql_batch_completed	2021-11-22 14:15:00....
sql_batch_completed	2021-11-22 14:15:01....
sql_batch_completed	2021-11-22 14:15:04....

Stop session after finishing.

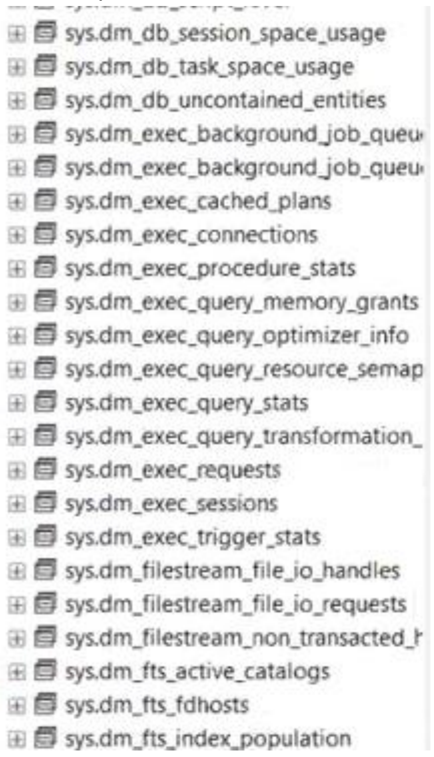
2) Logs



```
Management
  Policy Management
  Data Collection
  Resource Governor
  Extended Events
  Maintenance Plans
  SQL Server Logs
    Current - 11/27/2021 6:25:00 PM
    Archive #1 - 11/27/2021 1:36:00 AM
    Archive #2 - 11/26/2021 12:52:00 AM
    Archive #3 - 11/24/2021 11:24:00 PM
    Archive #4 - 11/24/2021 2:49:00 AM
    Archive #5 - 11/23/2021 12:07:00 AM
    Archive #6 - 11/22/2021 12:26:00 AM
```

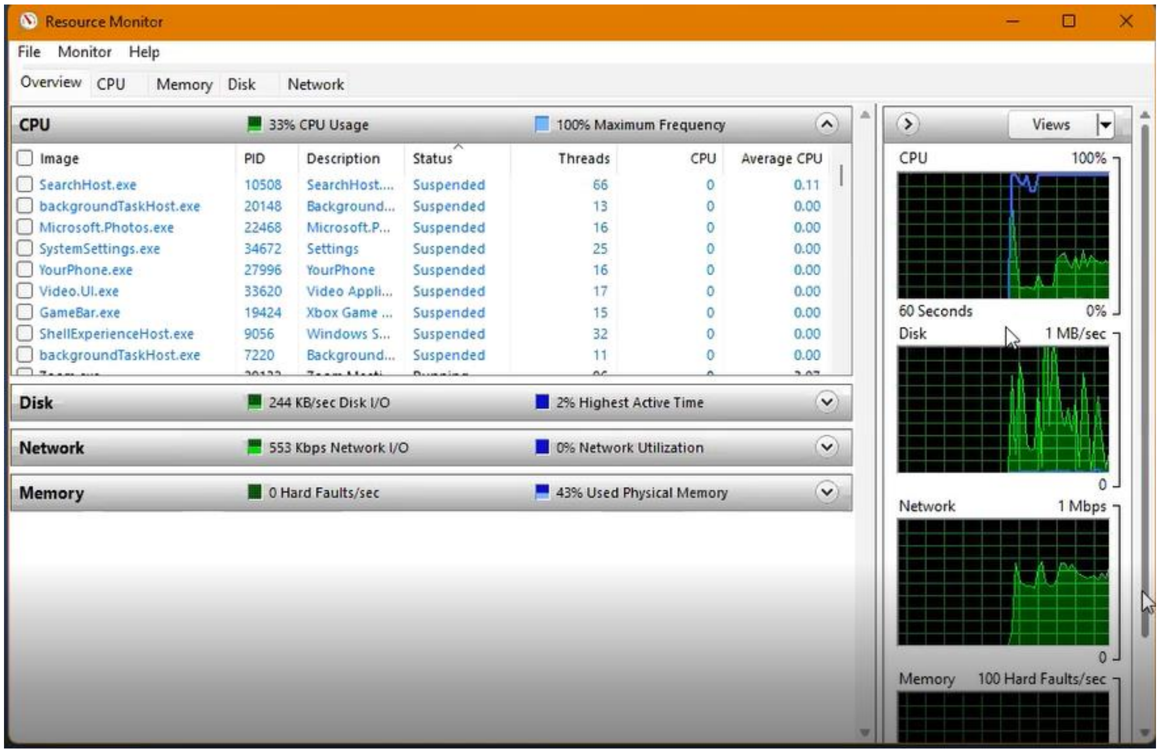
Use DBCC queries to turn on flag for certain events
Language : DBCC TRACEON(error_code, -1);
So that when the certain error happens, the log will catch it.

3) DMV
"Dynamic Management Views"
DMV is a collection of views in the masters database, and will tell you all the current status of the system. Start with sys.XXX



```
sys.dm_db_session_space_usage
sys.dm_db_task_space_usage
sys.dm_db_uncontained_entities
sys.dm_exec_background_job_queu
sys.dm_exec_background_job_queu
sys.dm_exec_cached_plans
sys.dm_exec_connections
sys.dm_exec_procedure_stats
sys.dm_exec_query_memory_grants
sys.dm_exec_query_optimizer_info
sys.dm_exec_query_resource_semap
sys.dm_exec_query_stats
sys.dm_exec_query_transformation_
sys.dm_exec_requests
sys.dm_exec_sessions
sys.dm_exec_trigger_stats
sys.dm_filestream_file_io_handles
sys.dm_filestream_file_io_requests
sys.dm_filestream_non_transacted_h
sys.dm_fts_active_catalogs
sys.dm_fts_fdhosts
sys.dm_fts_index_population
```

4) Resource Monitor (from Windows System)



Resource Monitor

File Monitor Help

Overview CPU Memory Disk Network

CPU 🟩 33% CPU Usage 🟦 100% Maximum Frequency

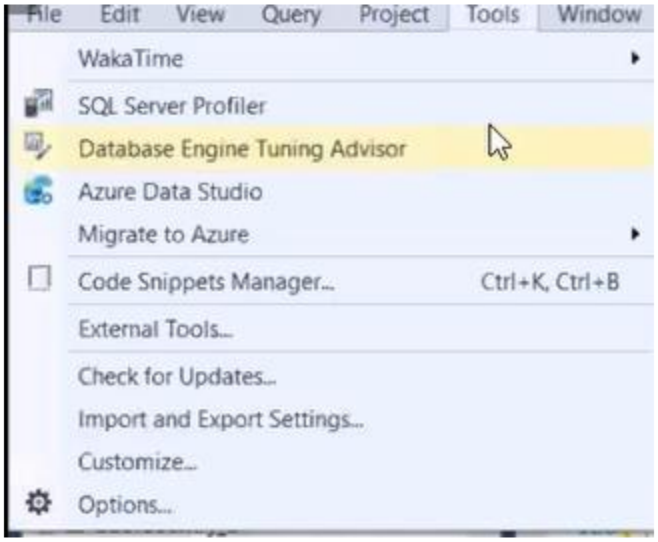
Image	PID	Description	Status	Threads	CPU	Average CPU
SearchHost.exe	10508	SearchHost....	Suspended	66	0	0.11
backgroundTaskHost.exe	20148	Background...	Suspended	13	0	0.00
Microsoft.Photos.exe	22468	Microsoft.P...	Suspended	16	0	0.00
SystemSettings.exe	34672	Settings	Suspended	25	0	0.00
YourPhone.exe	27996	YourPhone	Suspended	16	0	0.00
Video.UI.exe	33620	Video Appli...	Suspended	17	0	0.00
GameBar.exe	19424	Xbox Game ...	Suspended	15	0	0.00
ShellExperienceHost.exe	9056	Windows S...	Suspended	32	0	0.00
backgroundTaskHost.exe	7220	Background...	Suspended	11	0	0.00

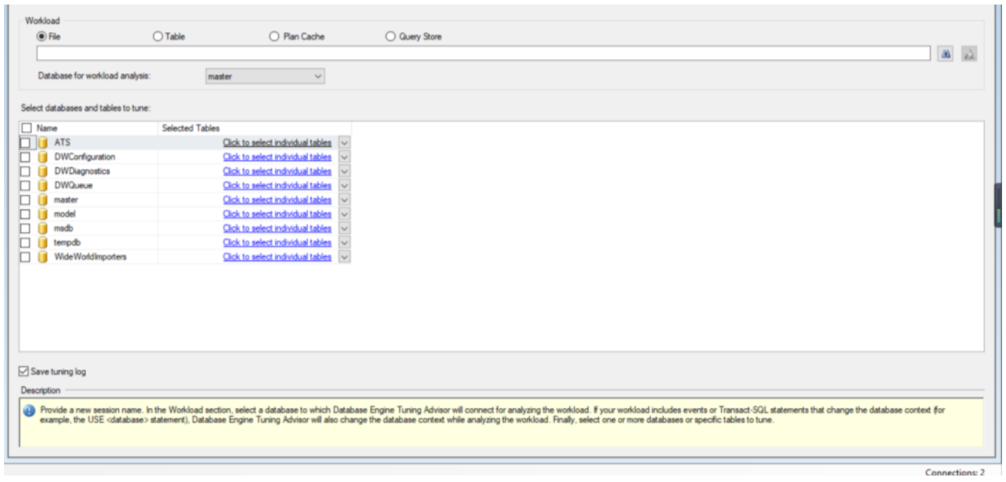
Disk 🟩 244 KB/sec Disk I/O 🟦 2% Highest Active Time

Network 🟩 553 Kbps Network I/O 🟦 0% Network Utilization

Memory 🟩 0 Hard Faults/sec 🟦 43% Used Physical Memory

5) Tuning Advisor


File Edit View Query Project Tools Window
WakaTime
SQL Server Profiler
Database Engine Tuning Advisor
Azure Data Studio
Migrate to Azure
Code Snippets Manager... Ctrl+K, Ctrl+B
External Tools...
Check for Updates...
Import and Export Settings...
Customize...
Options...



Workload

[✓ File] Table Plan Cache Query Store

Database for workload analysis: master

Select databases and tables to tune:

Name	Selected Tables
ATS	Click to select individual tables
DWConfiguration	Click to select individual tables
DWDiagnostics	Click to select individual tables
DWQueue	Click to select individual tables
master	Click to select individual tables
model	Click to select individual tables
msdb	Click to select individual tables
tempdb	Click to select individual tables
WideWorldImporters	Click to select individual tables

☑ Save tuning log

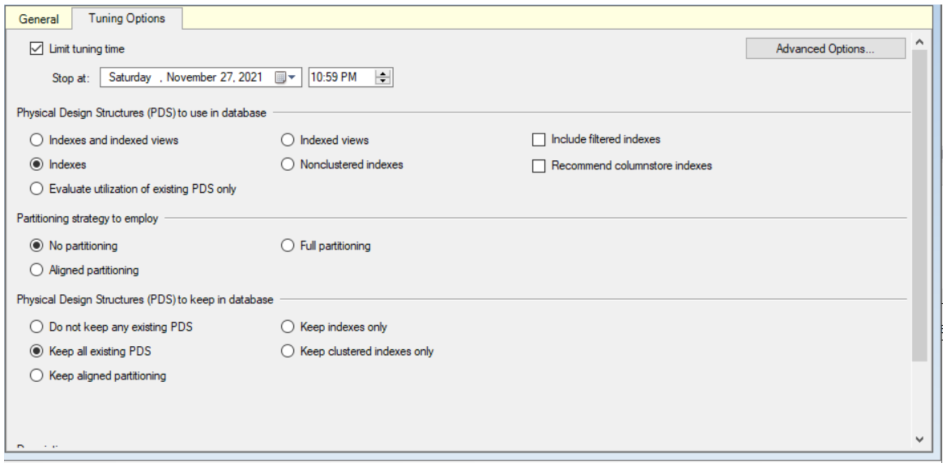
Description

Provide a new session name. In the Workload section, select a database to which Database Engine Tuning Advisor will connect for analyzing the workload. If your workload includes events or Transact-SQL statements that change the database context (for example, the USE <database> statement), Database Engine Tuning Advisor will also change the database context while analyzing the workload. Finally, select one or more databases or specific tables to tune.

Connections: 2

The "Plan Cache" is all the execution plans from stored procedure in the db. "Query Store" stores several queries ran in the past.

Go to tuning options, and you can specify what kind of index/views you would like to use. Usually will choose "Keep all existing PDS" for the last tab.



☑ Limit tuning time

Stop at: Saturday , November 27, 2021 10:59 PM

Physical Design Structures (PDS) to use in database

Indexes and indexed views / [✓ Indexes] / Evaluate utilization of existing PDS only / Indexed views / Nonclustered indexes / ☐ Include filtered indexes / ☐ Recommend columnstore indexes

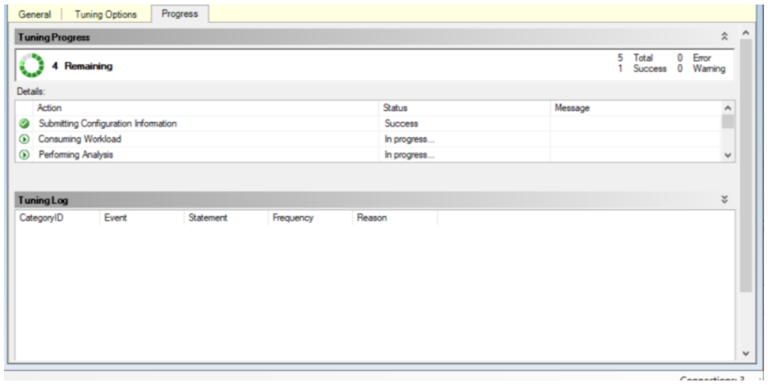
Partitioning strategy to employ

[✓ No partitioning] / Aligned partitioning / Full partitioning

Physical Design Structures (PDS) to keep in database

Do not keep any existing PDS / [✓ Keep all existing PDS] / Keep aligned partitioning / Keep indexes only / Keep clustered indexes only

Click "start analyze", then will get a report:



Tuning Progress

4 Remaining

5 Total 0 Error
1 Success 0 Warning

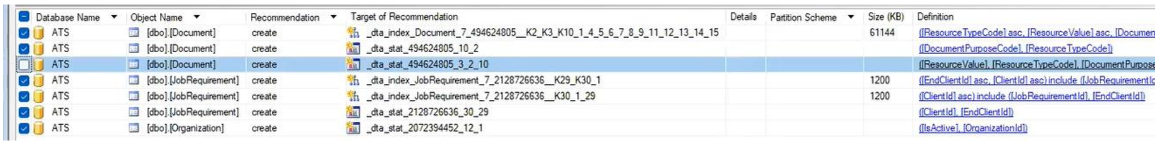
Details:

Action	Status	Message
Submitting Configuration Information	Success	
Consuming Workload	In progress...	
Performing Analysis	In progress...	

Tuning Log

CategoryID	Event	Statement	Frequency	Reason

The report will give partition and index recommendations, so one can automatically implement the recommendation via the "action" tab in the tuning advisor.

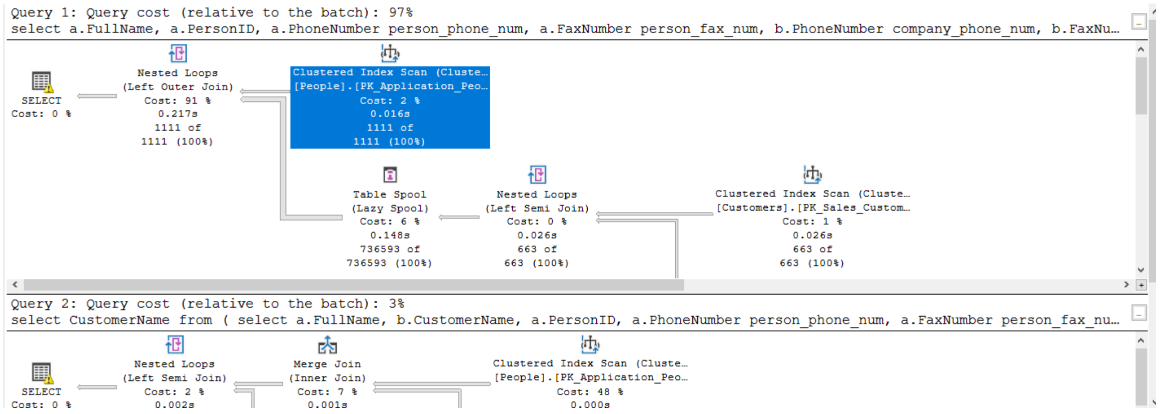


Database Name	Object Name	Recommendation	Target of Recommendation	Details	Partition Scheme	Size (KB)	Definition
✓ ATS	[dbo].[Document]	create	_dta_index_Document_7_494624805__K2_K3_K10_1_4_5_6_7_8_9_11_12_13_14_15			61144	([ResourceTypeCode] asc, [ResourceValue] asc, [Documen
✓ ATS	[dbo].[Document]	create	_dta_stat_494624805_10_2				([DocumentPurposeCode], [ResourceTypeCode])
ATS	[dbo].[Document]	create	_dta_stat_494624805_3_2_10				([ResourceValue], [ResourceTypeCode], [DocumentPurpose
✓ ATS	[dbo].[JobRequirement]	create	_dta_index_JobRequirement_7_2128726636__K29_K30_1			1200	([EndClientId] asc, [ClientId] asc) include ([JobRequirementI
✓ ATS	[dbo].[JobRequirement]	create	_dta_index_JobRequirement_7_2128726636__K30_1_29			1200	([ClientId] asc) include ([JobRequirementId], [EndClientId])
✓ ATS	[dbo].[JobRequirement]	create	_dta_stat_2128726636_30_29				([ClientId], [EndClientId])
✓ ATS	[dbo].[Organization]	create	_dta_stat_2072394452_12_1				([IsActive], [OrganizationId])

6) Execution Plan

If click this icon before running multiple queries, there will be a window called execution plan, which gives information about the workflow of each query and how much resources that query cost in terms of percentage out of all queries.

- Can be used to compare two queries that do the same thing. The query with lower percentage will have higher efficiency, thus is more desired.
- Can be used to see if an index is missing. If in the node of each execution plan, there's something called "Table Scan", it means there's an index missing.



```
Query 1: Query cost (relative to the batch): 97%
select a.FullName, a.PersonID, a.PhoneNumber person_phone_num, a.FaxNumber person_fax_num, b.PhoneNumber company_phone_num, b.FaxNu…
```

```
Query 2: Query cost (relative to the batch): 3%
select CustomerName from ( select a.FullName, b.CustomerName, a.PersonID, a.PhoneNumber person_phone_num, a.FaxNumber person_fax_nu…
```

Assignments 30 - 32 are group assignments.

30. Write a short essay talking about a scenario: Good news everyone! We (Wide World Importers) just brought out a small company called "Adventure works"! Now that bike shop is our sub-company. The first thing of all works pending would be to merge the user logon information, person information (including emails, phone numbers) and products (of course, add category, colors) to WWI database. Include screenshot, mapping and query.

    We first inserted data from Person.Person table in AdventureWorks into Application.People in WWI by the following codes:

```
INSERT INTO WideWorldImporters.Application.People
(FullName, PreferredName, IsPermittedToLogon, LogonName, IsExternalLogonProvider,
HashedPassword,
IsSystemUser, IsEmployee, IsSalesperson, PhoneNumber, EmailAddress, CustomFields,
LastEditedBy)
SELECT CONCAT(p.FirstName, p.MiddleName, p.LastName) AS FullName,
p.FirstName AS PreferredName,
CASE WHEN e.LoginID IS NOT NULL THEN 1 ELSE 0 END AS IsPermittedToLogon,
ISNULL(e.LoginID, 'NO LOGON') AS LogonName,
0 AS IsExternalLogonProvider,
CONVERT(varbinary(max), pw.PasswordHash) AS HashedPassword,
CASE WHEN pw.PasswordHash IS NOT NULL THEN 1 ELSE 0 END AS IsSystemUser,
CASE WHEN e.JobTitle IS NOT NULL THEN 1 ELSE 0 END AS IsEmployee,
CASE WHEN e.JobTitle LIKE '%Sales%' THEN 1 ELSE 0 END AS IsSalesperson,
pp.PhoneNumber AS PhoneNumber,
email.EmailAddress AS EmailAddress,
CONCAT('{ "OtherLanguages": [] ,"HireDate":"', e.HireDate, '","Title":"',
e.JobTitle, '"}') AS CustomFields,
1 AS LastEditedBy
FROM AdventureWorks2019.Person.Person p
LEFT JOIN AdventureWorks2019.HumanResources.Employee e
ON p.BusinessEntityID = e.BusinessEntityID
LEFT JOIN AdventureWorks2019.Person.Password pw
ON p.BusinessEntityID = pw.BusinessEntityID
LEFT JOIN AdventureWorks2019.Person.PersonPhone pp
ON p.BusinessEntityID = pp.BusinessEntityID
LEFT JOIN AdventureWorks2019.Person.EmailAddress email
       ON p.BusinessEntityID = email.BusinessEntityID;
```

We then insert data from Production.Product and Production.ProductCategory in AdventureWorks into Warehouse.Colors, Purchasing.Suppliers, and Warehouse.StockGroups in WWI by the following codes:

```
INSERT INTO WideWorldImporters.Warehouse.Colors
(ColorName, LastEditedBy)
SELECT DISTINCT Color AS ColorName, 1 AS LastEditedBy
FROM AdventureWorks2019.Production.Product p
WHERE p.Color IS NOT NULL AND NOT EXISTS
(SELECT * FROM WideWorldImporters.Warehouse.Colors c
WHERE c.ColorName = p.Color COLLATE Latin1_General_100_CI_AS);

INSERT INTO WideWorldImporters.Purchasing.Suppliers
(SupplierName, SupplierCategoryID, PrimaryContactPersonID,
AlternateContactPersonID, DeliveryCityID,
PostalCityID, PaymentDays, BankAccountNumber, PhoneNumber, FaxNumber, WebsiteURL,
DeliveryAddressLine1, DeliveryPostalCode,
PostalAddressLine1, PostalPostalCode, LastEditedBy)
SELECT v.Name AS SupplierName,
1 AS SupplierCategoryID, 1 AS PrimaryContactPersonID, 1 AS
AlternateContactPersonID, 1 AS DeliveryCityID, 1 AS PostalCityID,
0 AS PaymentDays, v.AccountNumber AS BankAccountNumber, '' AS PhoneNumber, ''
[FaxNumber], '' [WebsiteURL], '' [DeliveryAddressLine1],
'' [DeliveryPostalCode], '' [PostalAddressLine1], '' [PostalPostalCode], 1
[LastEditedBy]
FROM AdventureWorks2019.Purchasing.Vendor v
WHERE NOT EXISTS
(SELECT * FROM WideWorldImporters.Purchasing.Suppliers s
WHERE s.SupplierName = v.Name COLLATE Latin1_General_100_CI_AS);

INSERT INTO WideWorldImporters.Warehouse.StockGroups
(StockGroupName, LastEditedBy)
SELECT pc.Name AS StockGroupName, 1 AS LastEditedBy
FROM AdventureWorks2019.Production.ProductCategory pc
WHERE NOT EXISTS
(SELECT * FROM WideWorldImporters.Warehouse.StockGroups
       WHERE StockGroupName = pc.Name COLLATE Latin1_General_100_CI_AS);
```

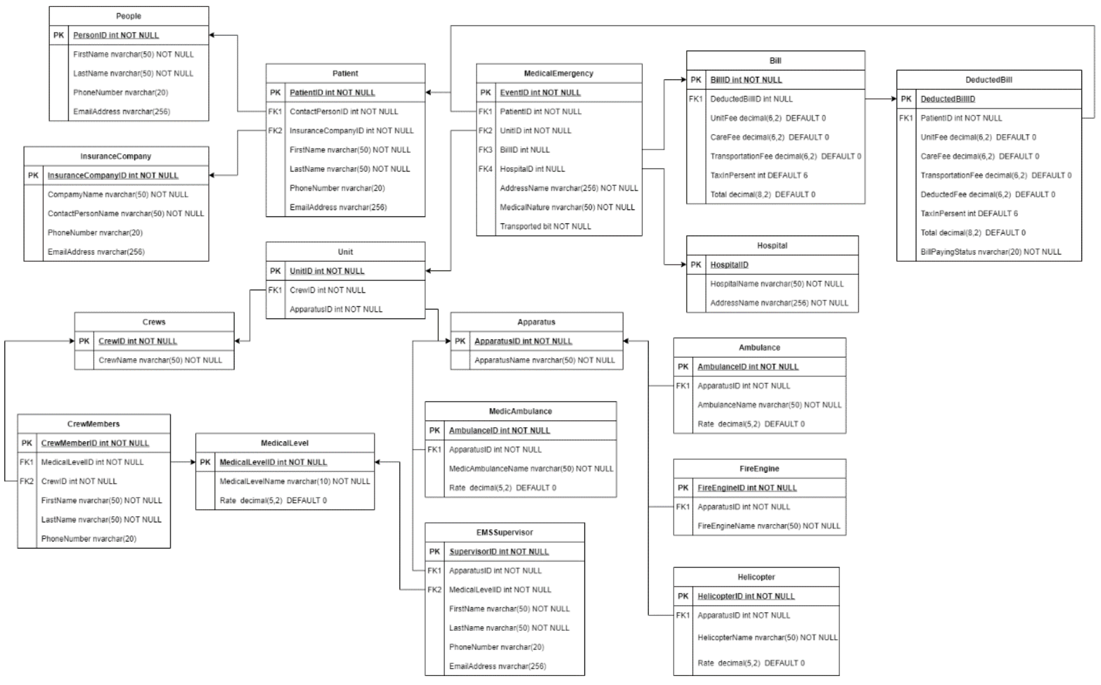
In the next step, we created a temporal table to save the query results from joined multiple tables related to product information in AdventureWorks and inserted them into "Warehouse.StockItems" in WWI by the following codes:

```sql
SELECT DISTINCT p.Name AS StockItemName,
s.SupplierID AS SupplierID,
c.ColorID AS ColorID,
7 AS UnitPackageID,
7 AS OuterPackageID,
p.Size AS Size,
pv.AverageLeadTime AS LeadTimeDays,
1 As QuantityPerOuter,
0 AS IsChillerStock,
6.0 AS TaxRate,
p.ListPrice AS UnitPrice,
pv.StandardPrice AS RecommendedRetailPrice,
ISNULL(p.Weight,0) AS TypicalWeightPerUnit,
pd.Description AS MarketingComments,
pp.LargePhoto AS Photo,
1 AS LastEditedBy,
ROW_NUMBER() OVER(PARTITION BY p.ProductID ORDER BY p.Name) AS Row
INTO #Temp
FROM AdventureWorks2019.Production.Product p
INNER JOIN AdventureWorks2019.Purchasing.ProductVendor pv
ON p.ProductID = pv.ProductID
INNER JOIN AdventureWorks2019.Purchasing.Vendor v
ON pv.BusinessEntityID = v.BusinessEntityID
INNER JOIN WideWorldImporters.Purchasing.Suppliers s
ON v.Name = s.SupplierName COLLATE Latin1_General_100_CI_AS
INNER JOIN AdventureWorks2019.Production.ProductModel pm
ON p.ProductModelID = pm.ProductModelID
INNER JOIN AdventureWorks2019.Production.ProductModelProductDescriptionCulture
pmpdc
ON pm.ProductModelID = pmpdc.ProductModelID
INNER JOIN AdventureWorks2019.Production.ProductDescription pd
ON pmpdc.ProductDescriptionID = pd.ProductDescriptionID
INNER JOIN AdventureWorks2019.Production.ProductProductPhoto ppp
ON p.ProductID = ppp.ProductID
INNER JOIN AdventureWorks2019.Production.ProductPhoto pp
ON ppp.ProductPhotoID = pp.ProductPhotoID
INNER JOIN WideWorldImporters.Warehouse.Colors c
ON p.Color = c.ColorName COLLATE Latin1_General_100_CI_AS
WHERE NOT EXISTS
(SELECT * FROM WideWorldImporters.Warehouse.StockItems si
WHERE si.StockItemName = p.Name COLLATE Latin1_General_100_CI_AS);

INSERT INTO WideWorldImporters.Warehouse.StockItems
(StockItemName, SupplierID, ColorID, UnitPackageID, OuterPackageID, [Size],
LeadTimeDays, QuantityPerOuter, IsChillerStock,
TaxRate, UnitPrice, [RecommendedRetailPrice], TypicalWeightPerUnit,
[MarketingComments], [Photo], LastEditedBy)
SELECT
CONCAT(StockItemName, Row) AS StockItemName, SupplierID, ColorID, UnitPackageID,
OuterPackageID, Size, LeadTimeDays, QuantityPerOuter,
IsChillerStock, TaxRate, UnitPrice, RecommendedRetailPrice, TypicalWeightPerUnit,
MarketingComments, Photo, LastEditedBy
       FROM #Temp;
```

Finally, we insert information from "Warehouse.StockItems" and "Warehouse.StockGroups" to the joined table, "Warehouse.StockItemStockGroups" by the following codes:

```
INSERT INTO WideWorldImporters.Warehouse.StockItemStockGroups
(StockItemID, StockGroupID, LastEditedBy)
SELECT si.StockItemID, ps.ProductCategoryID AS StockGroupID, 1 [LastEditedBy]
FROM AdventureWorks2019.Production.Product p
INNER JOIN AdventureWorks2019.Production.ProductSubcategory ps
ON p.ProductSubcategoryID = ps.ProductSubcategoryID
INNER JOIN #Temp ON p.Name = #Temp.StockItemName
INNER JOIN WideWorldImporters.Warehouse.StockItems si
       ON CONCAT(#Temp.StockItemName, #Temp.Row) = si.StockItemName COLLATE
       Latin1_General_100_CI_AS;
```

31. Database Design: OLTP db design request for EMS business: when people call 911 for medical emergency, 911 will dispatch UNITs to the given address. A UNIT means a crew on an apparatus (Fire Engine, Ambulance, Medic Ambulance, Helicopter, EMS supervisor). A crew member would have a medical level (EMR, EMT, A-EMT, Medic). All the treatments provided on scene are free. If the patient needs to be transported, that's where the bill comes in. A bill consists of Units dispatched (Fire Engine and EMS Supervisor are free), crew members provided care (EMRs and EMTs are free), Transported miles from the scene to the hospital (Helicopters have a much higher rate, as you can image) and tax (Tax rate is 6%). Bill should be sent to the patient insurance company first. If there is a deductible, we send the unpaid bill to the patient only. Don't forget about patient information, medical nature and bill paying status.



People
PK PersonID int NOT NULL
FirstName nvarchar(50) NOT NULL
LastName nvarchar(50) NOT NULL
PhoneNumber nvarchar(20)
EmailAddress nvarchar(256)
Patient
PK PatientID int NOT NULL
FK1 ContactPersonID int NOT NULL
FK2 InsuranceCompanyID int NOT NULL
FirstName nvarchar(50) NOT NULL
LastName nvarchar(50) NOT NULL
PhoneNumber nvarchar(20)
EmailAddress nvarchar(256)
MedicalEmergency
PK EventID int NOT NULL
FK1 PatientID int NOT NULL
FK2 UnitID int NOT NULL
FK3 BillID int NULL
FK4 HospitalD int NULL
AddressName nvarchar(256) NOT NULL
MedicalNature nvarchar(50) NOT NULL
Transported bit NOT NULL
Bill
PK BillID int NOT NULL
FK1 DeductedBillID int NULL
UnitFee decimal(6,2) DEFAULT 0
CareFee decimal(6,2) DEFAULT 0
TransportationFee decimal(6,2) DEFAULT 0
TaxInPersent int DEFAULT 6
Total decimal(8,2) DEFAULT 0
DeductedBill
PK DeductedBillID
FK1 PatientID int NOT NULL
UnitFee decimal(6,2) DEFAULT 0
CareFee decimal(6,2) DEFAULT 0
TransportationFee decimal(6,2) DEFAULT 0
DeductedFee decimal(6,2) DEFAULT 0
TaxInPersent int DEFAULT 6
Total decimal(8,2) DEFAULT 0
BillPayingStatus nvarchar(20) NOT NULL
InsuranceCompany
PK InsuranceCompanyID int NOT NULL
CompamyName nvarchar(50) NOT NULL
ContactPersonName nvarchar(50) NOT NULL
PhoneNumber nvarchar(20)
EmailAddress nvarchar(256)
Unit
PK UnitID int NOT NULL
FK1 CrewID int NOT NULL
ApparatusID int NOT NULL
Hospital
PK HospitalID
HospitalName nvarchar(50) NOT NULL
AddressName nvarchar(256) NOT NULL
Crews
PK CrewID int NOT NULL
CrewName nvarchar(50) NOT NULL
Apparatus
PK ApparatusID int NOT NULL
ApparatusName nvarchar(50) NOT NULL
Ambulance
PK AmbulanceID int NOT NULL
FK1 ApparatusID int NOT NULL
AmbulanceName nvarchar(50) NOT NULL
Rate decimal(5,2) DEFAULT 0
MedicAmbulance
PK AmbulanceID int NOT NULL
FK1 ApparatusID int NOT NULL
MedicAmbulanceName nvarchar(50) NOT NULL
Rate decimal(5,2) DEFAULT 0
CrewMembers
PK CrewMemberID int NOT NULL
FK1 MedicalLevelID int NOT NULL
FK2 CrewID int NOT NULL
FirstName nvarchar(50) NOT NULL
LastName nvarchar(50) NOT NULL
PhoneNumber nvarchar(20)
MedicalLevel
PK MedicalLevelID int NOT NULL
MedicalLevelName nvarchar(10) NOT NULL
Rate decimal(5,2) DEFAULT 0
FireEngine
PK FireEngineID int NOT NULL
FK1 ApparatusID int NOT NULL
FireEngineName nvarchar(50) NOT NULL
EMSSupervisor
PK SupervisorID int NOT NULL
FK1 ApparatusID int NOT NULL
FK2 MedicalLevelID int NOT NULL
FirstName nvarchar(50) NOT NULL
LastName nvarchar(50) NOT NULL
PhoneNumber nvarchar(20)
EmailAddress nvarchar(256)
Helicopter
PK HelicopterID int NOT NULL
FK1 ApparatusID int NOT NULL
HelicopterName nvarchar(50) NOT NULL
Rate decimal(5,2) DEFAULT 0

32. Remember the discussion about those two databases from the class, also remember, those data models are not perfect. You can always add new columns (but not alter or drop columns) to any tables. Suggesting adding Ingested DateTime and Surrogate Key columns. Study the Wide World Importers DW. Think the integration schema is the ODS. Come up with a TSQL Stored Procedure driven solution to move the data from WWI database to ODS, and then from the ODS to the fact tables and dimension tables. By the way, WWI DW is a galaxy schema db. Requirements:
    a. Luckly, we only start with 1 fact: Purchase. Other facts can be ignored for now.
    b. Add a new dimension: Country of Manufacture. It should be given on top of Stock Items.
    c. Write script(s) and stored procedure(s) for the entire ETL from WWI db to DW.

```sql
ALTER TABLE WideWorldImportersDW.Dimension.[Stock Item]
ADD [Country of Manufacture] NVARCHAR(20)

UPDATE WideWorldImportersDW.Dimension.[Stock Item]
SET [Country of Manufacture] =
JSON_VALUE(SI.CustomFields,'$.CountryOfManufacture')
FROM WideWorldImporters.Warehouse.StockItems AS SI
WHERE [Stock Item Key] = SI.StockItemID

SELECT [Country of Manufacture] FROM WideWorldImportersDW.Dimension.[Stock Item]

CREATE PROCEDURE dbo.ExtractOrder
AS
       SELECT
       C.DeliveryCityID,
       O.CustomerID,
       OL.StockItemID ,
       O.OrderDate,
       CONVERT(DATE,O.PickingCompletedWhen) AS [Picked Date Key],
       O.SalespersonPersonID,
       O.PickedByPersonID,
       O.OrderID ,
       O.BackorderOrderID ,
       SI.StockItemName,
       PT.PackageTypeName,
       OL.Quantity,
       OL.UnitPrice,
       OL.TaxRate,
       IL.TaxAmount
       FROM WideWorldImporters.Sales.Orders AS O
       JOIN WideWorldImporters.Sales.OrderLines AS OL
       ON O.OrderID = OL.OrderID
       JOIN WideWorldImporters.Sales.Invoices AS I
       ON I.OrderID = O.OrderID
       JOIN WideWorldImporters.Sales.InvoiceLines AS IL
       ON IL.InvoiceID = I.InvoiceID AND IL.StockItemID = OL.StockItemID
       JOIN WideWorldImporters.Warehouse.StockItems AS SI
       ON SI.StockItemID = OL.StockItemID
       JOIN WideWorldImporters.Warehouse.PackageTypes AS PT
       ON PT.PackageTypeID = OL.PackageTypeID
       JOIN WideWorldImporters.Sales.Customers AS C
       ON C.CustomerID = O.CustomerID
```

```
GO

CREATE TABLE WideWorldImportersDW.Integration.ExtractOrder_Staging(
       DeliveryCityID INT,
       CustomerID INT,
       StockItemID INT ,
       OrderDate DATE ,
       [Picked Date Key] DATE ,
       SalespersonPersonID INT ,
       PickedByPersonID INT ,
       OrderID INT,
       BackorderOrderID INT ,
       StockItemName NVARCHAR(MAX),
       PackageTypeName NVARCHAR(50),
       Quantity INT,
       UnitPrice DECIMAL(18,2),
       TaxRate DECIMAL(18,3),
       TaxAmount DECIMAL(18,2)
);

INSERT INTO WideWorldImportersDW.Integration.ExtractOrder_Staging
    EXEC dbo.ExtractOrder ;

CREATE PROCEDURE dbo.TrasformOrder
AS
        SELECT
        DeliveryCityID,
        CustomerID,
        StockItemID,
        OrderDate,
        [Picked Date Key],
        SalespersonPersonID,
        PickedByPersonID,
        OrderID,
        BackorderOrderID,
        StockItemName,
        PackageTypeName,
        Quantity,
        UnitPrice,
        TaxRate,
        Quantity*UnitPrice  AS [Total Excluding Tax],
        TaxAmount,
        Quantity*UnitPrice + TaxAmount AS [Total Including Tax]
        FROM WideWorldImportersDW.Integration.ExtractOrder_Staging

GO

CREATE TABLE WideWorldImportersDW.Integration.TransformOrder_Staging(
       DeliveryCityID INT,
       CustomerID INT,
       StockItemID INT ,
       OrderDate DATE ,
       [Picked Date Key] DATE ,
       SalespersonPersonID INT ,
       PickedByPersonID INT ,
       OrderID INT,
       BackorderOrderID INT ,
	StockItemName NVARCHAR(MAX),
	PackageTypeName NVARCHAR(50),
	Quantity INT,
	UnitPrice DECIMAL(18,2),
	TaxRate DECIMAL(18,3),
	[Total Excluding Tax] DECIMAL(18,3),
	TaxAmount DECIMAL(18,2),
	[Total Including Tax] DECIMAL(18,3)
);

INSERT INTO  WideWorldImportersDW.Integration.TransformOrder_Staging
	EXEC dbo.TrasformOrder;

DROP TABLE WideWorldImportersDW.Integration.ExtractOrder_Staging;

CREATE PROCEDURE dbo.LoadOrder
AS
	INSERT INTO WideWorldImportersDW.Fact.[Order](
	 [City Key],
	 [Customer Key],
	 [Stock Item Key],
	 [Order Date Key],
	 [Picked Date Key],
	 [Salesperson Key],
	 [Picker Key],
	 [WWI Order ID],
	 [WWI Backorder ID],
	 [Description],
	 [Package],
	 Quantity,
	 [Unit Price],
	 [Tax Rate],
	 [Total Excluding Tax],
	 [Tax Amount],
	 [Total Including Tax],
	 [Lineage Key])

	SELECT
	 City.[City Key],
	 ISNULL(C.[Customer Key],0) AS [Customer Key],
	 SI.[Stock Item Key],
	 OrderDate AS [Order Date Key],
	 [Picked Date Key],
	 E.[Employee Key] AS [Salesperson Key],
	 EE.[Employee Key] AS [Picker Key],
	 OrderID AS [WWI Order ID],
	 BackorderOrderID AS [WWI Backorder ID],
	 StockItemName AS [Description],
	 PackageTypeName AS [Package],
	 Quantity,
	 UnitPrice AS [Unit Price],
	 TaxRate AS [Tax Rate],
	 [Total Excluding Tax],
	 TaxAmount AS [Tax Amount],
	 [Total Including Tax],
	 9
	FROM WideWorldImportersDW.Integration.TransformOrder_Staging    AS A
	LEFT JOIN WideWorldImportersDW.Dimension.Customer AS C
```

```
       ON A.CustomerID = C.[WWI Customer ID] AND C.[Valid To]='9999-12-31
23:59:59.9999999'
       LEFT JOIN WideWorldImportersDW.Dimension.[Stock Item] AS SI
       ON SI.[WWI Stock Item ID] = A.StockItemID AND SI.[Valid To] = '9999-12-31
23:59:59.9999999'
       LEFT JOIN WideWorldImportersDW.Dimension.Employee AS E
       ON E.[WWI Employee ID] = A.SalespersonPersonID AND E.[Valid To] = '9999-12-
31 23:59:59.9999999'
       LEFT JOIN WideWorldImportersDW.Dimension.Employee AS EE
       ON EE.[WWI Employee ID] = A.PickedByPersonID AND EE.[Valid To] = '9999-12-
31 23:59:59.9999999'
       LEFT JOIN WideWorldImportersDW.Dimension.City AS City
       ON City.[WWI City ID] = A.DeliveryCityID AND City.[Valid To] = '9999-12-31
23:59:59.9999999'

EXEC dbo.LoadOrder
       DROP TABLE WideWorldImportersDW.Integration.TransformOrder_Staging;
```