Amazon Superstore Analysis Report

SQL Project Report: Unveiling Insights from Amazon Superstore Data

This report leverages the power of SQL to delve into the Amazon Superstore dataset, uncovering hidden patterns and actionable insights to optimize sales, marketing, and inventory management. Through meticulous querying, data manipulation, and analysis, we expose key trends in customer behavior, product performance, and pricing strategies. Armed with these insights, we present concrete recommendations to propel Amazon's superstores to even greater heights.

Methodology

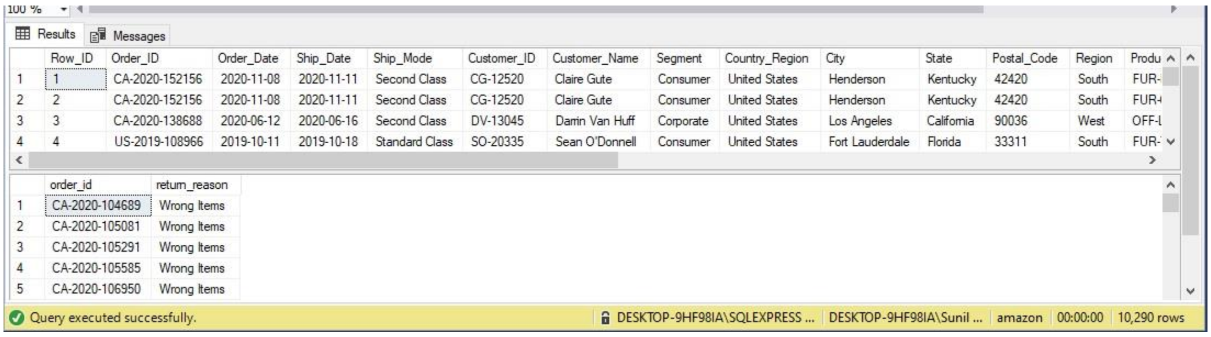
- Data Acquisition: We utilized the publicly available Amazon Superstore dataset, encompassing sales, customer, product, and geographical information.
- Data Cleaning and Preprocessing: We employed SQL functions and joins to address missing values, inconsistencies, and duplicate entries.
- Exploratory Data Analysis (EDA): We crafted SQL queries to generate descriptive statistics, identify outliers, and visualize data trends through aggregations and joins.
- Statistical Analysis: We used SQL functions and windowing techniques to calculate sales growth, customer segmentation metrics, and product performance measures.
- Advanced Analysis: We employed self-joins and subqueries to explore deeper relationships, such as customer lifetime value and basket analysis.

Key Findings (Focus on SQL-driven insights)

- Customer Segmentation: Using SQL functions like **`COUNTIF`** and **`GROUP BY`**, we identified distinct customer segments based on purchase frequency and average order value.
- Top-Selling Products: Through queries involving **`SUM`**, **GROUP BY** , **`ORDER BY`**, we pinpointed the top-selling products and categories, revealing hidden gems and potential opportunities.
- Price Sensitivity: Employing **SQL joins and windowing functions,** we analyzed the impact of pricing on sales across different product categories and customer segments.
- Promotional Effectiveness: Utilizing **`CASE` statements and conditional joins,Sub-Queries,CTE's,Views, transactions**, I evaluated the effectiveness of various promotional campaigns, identifying the most impactful strategies

This SQL-powered analysis has unlocked a treasure trove of insights into the inner workings of Amazon's superstores. By embracing the power of SQL queries, we have identified key areas for improvement and provided actionable recommendations to optimize sales, marketing, and inventory management. By continuously iterating and refining these SQL-driven strategies, Amazon can solidify its position as a leader in the e-commerce landscape.

```sql
use amazon;
select * from orders;
select * from returns;
```



100 %

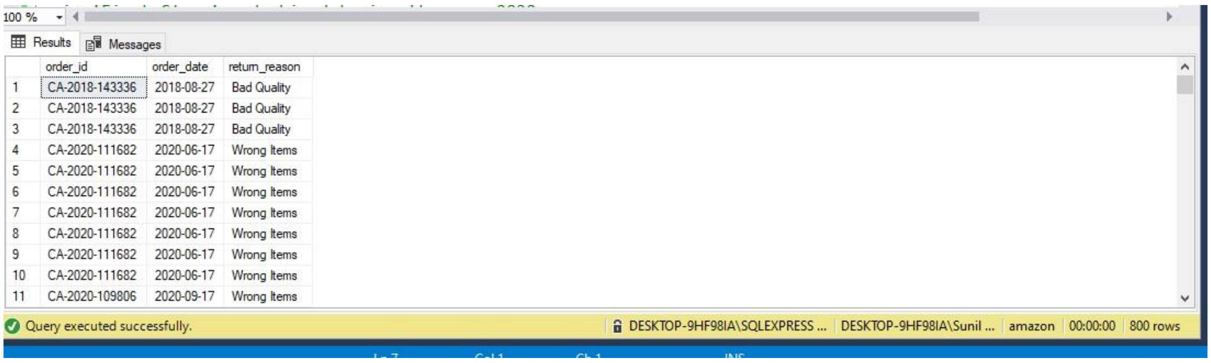
Results | Messages

	Row_ID	Order_ID	Order_Date	Ship_Date	Ship_Mode	Customer_ID	Customer_Name	Segment	Country_Region	City	State	Postal_Code	Region	Produ
1	1	CA-2020-152156	2020-11-08	2020-11-11	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	Kentucky	42420	South	FUR-
2	2	CA-2020-152156	2020-11-08	2020-11-11	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	Kentucky	42420	South	FUR-
3	3	CA-2020-138688	2020-06-12	2020-06-16	Second Class	DV-13045	Darrin Van Huff	Corporate	United States	Los Angeles	California	90036	West	OFF-l
4	4	US-2019-108966	2019-10-11	2019-10-18	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	Florida	33311	South	FUR-

	order_id	return_reason
1	CA-2020-104689	Wrong Items
2	CA-2020-105081	Wrong Items
3	CA-2020-105291	Wrong Items
4	CA-2020-105585	Wrong Items
5	CA-2020-106950	Wrong Items

Query executed successfully. | DESKTOP-9HF98IA\SQLEXPRESS ... | DESKTOP-9HF98IA\Sunil ... | amazon | 00:00:00 | 10,290 rows

```sql
--1.)Joining TWO TABLES RETURNS AND ORDERS that have columns such as
order id,order date and returned.

SELECT o.order_id,o.order_date, r.return_reason from orders o join
returns r on o.order_id = r.order_id;
```



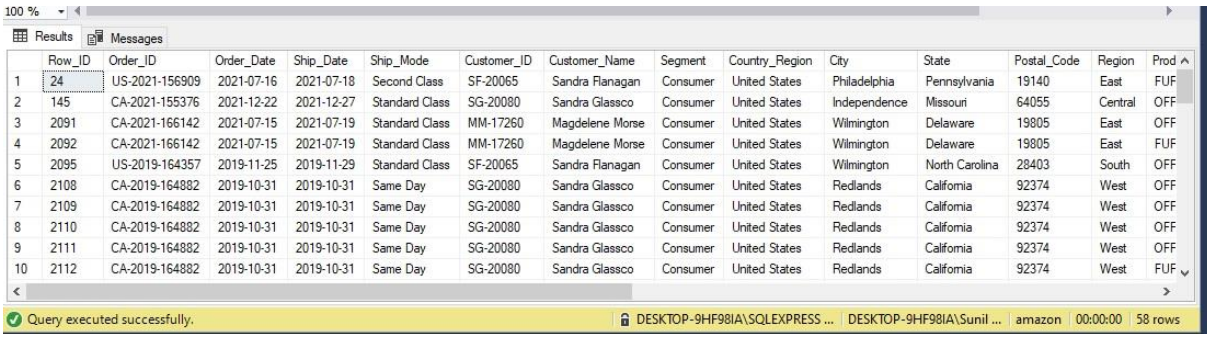
100 %

Results | Messages

	order_id	order_date	return_reason
1	CA-2018-143336	2018-08-27	Bad Quality
2	CA-2018-143336	2018-08-27	Bad Quality
3	CA-2018-143336	2018-08-27	Bad Quality
4	CA-2020-111682	2020-06-17	Wrong Items
5	CA-2020-111682	2020-06-17	Wrong Items
6	CA-2020-111682	2020-06-17	Wrong Items
7	CA-2020-111682	2020-06-17	Wrong Items
8	CA-2020-111682	2020-06-17	Wrong Items
9	CA-2020-111682	2020-06-17	Wrong Items
10	CA-2020-111682	2020-06-17	Wrong Items
11	CA-2020-109806	2020-09-17	Wrong Items

Query executed successfully. | DESKTOP-9HF98IA\SQLEXPRESS ... | DESKTOP-9HF98IA\Sunil ... | amazon | 00:00:00 | 800 rows

```sql
--2.)write a sql to get all the orders where customers name has "a"
as second character and "d" as fourth character

select * from orders where Customer_Name like '_a_d%';
```



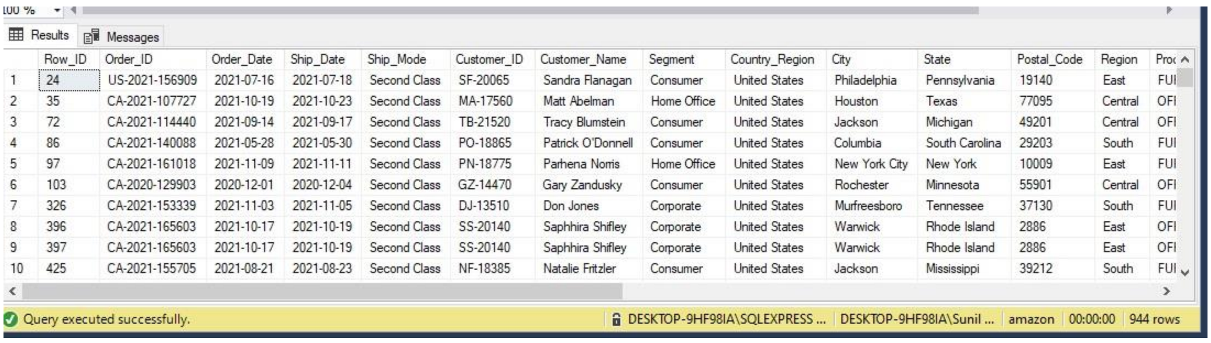
100 %

Results | Messages

	Row_ID	Order_ID	Order_Date	Ship_Date	Ship_Mode	Customer_ID	Customer_Name	Segment	Country_Region	City	State	Postal_Code	Region	Prod
1	24	US-2021-156909	2021-07-16	2021-07-18	Second Class	SF-20065	Sandra Flanagan	Consumer	United States	Philadelphia	Pennsylvania	19140	East	FUF
2	145	CA-2021-155376	2021-12-22	2021-12-27	Standard Class	SG-20080	Sandra Glassco	Consumer	United States	Independence	Missouri	64055	Central	OFF
3	2091	CA-2021-166142	2021-07-15	2021-07-19	Standard Class	MM-17260	Magdelene Morse	Consumer	United States	Wilmington	Delaware	19805	East	OFF
4	2092	CA-2021-166142	2021-07-15	2021-07-19	Standard Class	MM-17260	Magdelene Morse	Consumer	United States	Wilmington	Delaware	19805	East	FUF
5	2095	US-2019-164357	2019-11-25	2019-11-29	Standard Class	SF-20065	Sandra Flanagan	Consumer	United States	Wilmington	North Carolina	28403	South	OFF
6	2108	CA-2019-164882	2019-10-31	2019-10-31	Same Day	SG-20080	Sandra Glassco	Consumer	United States	Redlands	California	92374	West	OFF
7	2109	CA-2019-164882	2019-10-31	2019-10-31	Same Day	SG-20080	Sandra Glassco	Consumer	United States	Redlands	California	92374	West	OFF
8	2110	CA-2019-164882	2019-10-31	2019-10-31	Same Day	SG-20080	Sandra Glassco	Consumer	United States	Redlands	California	92374	West	OFF
9	2111	CA-2019-164882	2019-10-31	2019-10-31	Same Day	SG-20080	Sandra Glassco	Consumer	United States	Redlands	California	92374	West	OFF
10	2112	CA-2019-164882	2019-10-31	2019-10-31	Same Day	SG-20080	Sandra Glassco	Consumer	United States	Redlands	California	92374	West	FUF

Query executed successfully. | DESKTOP-9HF98IA\SQLEXPRESS ... | DESKTOP-9HF98IA\Sunil ... | amazon | 00:00:00 | 58 rows

```sql
--3.)write a query to get all the orders where ship_mode is neither in 'Standard Class' nor
--in 'First Class' and ship_date is after nov 2020

select * from orders where Ship_Mode not in ('Standard Class',
'First Class') and Ship_Date > '2020-11-30';
```



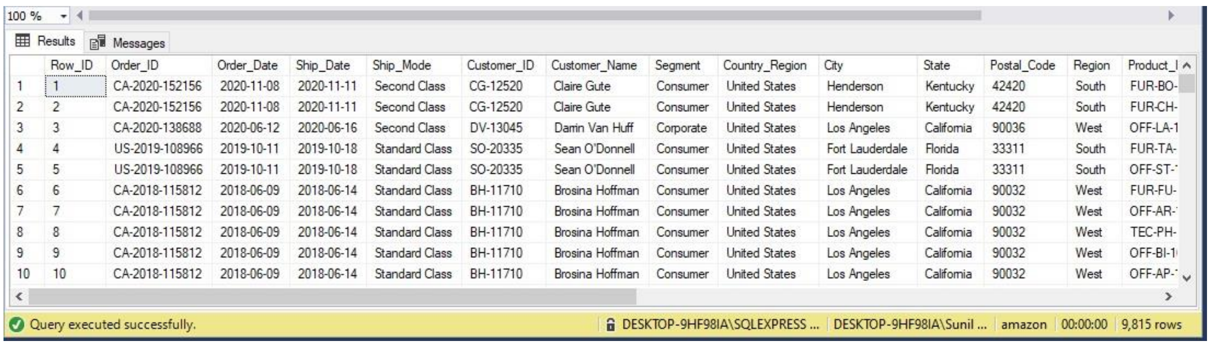
100 %

Results | Messages

	Row_ID	Order_ID	Order_Date	Ship_Date	Ship_Mode	Customer_ID	Customer_Name	Segment	Country_Region	City	State	Postal_Code	Region	Proc
1	24	US-2021-156909	2021-07-16	2021-07-18	Second Class	SF-20065	Sandra Flanagan	Consumer	United States	Philadelphia	Pennsylvania	19140	East	FUI
2	35	CA-2021-107727	2021-10-19	2021-10-23	Second Class	MA-17560	Matt Abelman	Home Office	United States	Houston	Texas	77095	Central	OFI
3	72	CA-2021-114440	2021-09-14	2021-09-17	Second Class	TB-21520	Tracy Blumstein	Consumer	United States	Jackson	Michigan	49201	Central	OFI
4	86	CA-2021-140088	2021-05-28	2021-05-30	Second Class	PO-18865	Patrick O'Donnell	Consumer	United States	Columbia	South Carolina	29203	South	FUI
5	97	CA-2021-161018	2021-11-09	2021-11-11	Second Class	PN-18775	Parhena Norris	Home Office	United States	New York City	New York	10009	East	FUI
6	103	CA-2020-129903	2020-12-01	2020-12-04	Second Class	GZ-14470	Gary Zandusky	Consumer	United States	Rochester	Minnesota	55901	Central	OFI
7	326	CA-2021-153339	2021-11-03	2021-11-05	Second Class	DJ-13510	Don Jones	Corporate	United States	Murfreesboro	Tennessee	37130	South	FUI
8	396	CA-2021-165603	2021-10-17	2021-10-19	Second Class	SS-20140	Saphhira Shifley	Corporate	United States	Warwick	Rhode Island	2886	East	OFI
9	397	CA-2021-165603	2021-10-17	2021-10-19	Second Class	SS-20140	Saphhira Shifley	Corporate	United States	Warwick	Rhode Island	2886	East	OFI
10	425	CA-2021-155705	2021-08-21	2021-08-23	Second Class	NF-18385	Natalie Fritzler	Consumer	United States	Jackson	Mississippi	39212	South	FUI

Query executed successfully. | DESKTOP-9HF98IA\SQLEXPRESS ... | DESKTOP-9HF98IA\Sunil ... | amazon | 00:00:00 | 944 rows

```sql
--4.)write a query to get all the orders where customer name neither start with "A" and nor ends with "n"

select * from orders where Customer_Name not like 'A%n';
```



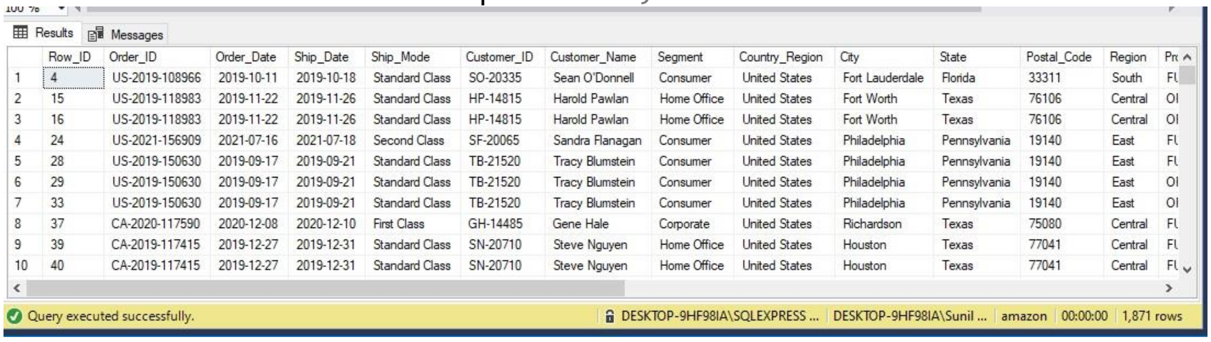
100 %

Results | Messages

	Row_ID	Order_ID	Order_Date	Ship_Date	Ship_Mode	Customer_ID	Customer_Name	Segment	Country_Region	City	State	Postal_Code	Region	Product_I
1	1	CA-2020-152156	2020-11-08	2020-11-11	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	Kentucky	42420	South	FUR-BO-
2	2	CA-2020-152156	2020-11-08	2020-11-11	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	Kentucky	42420	South	FUR-CH-
3	3	CA-2020-138688	2020-06-12	2020-06-16	Second Class	DV-13045	Darrin Van Huff	Corporate	United States	Los Angeles	California	90036	West	OFF-LA-1
4	4	US-2019-108966	2019-10-11	2019-10-18	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	Florida	33311	South	FUR-TA-
5	5	US-2019-108966	2019-10-11	2019-10-18	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	Florida	33311	South	OFF-ST-
6	6	CA-2018-115812	2018-06-09	2018-06-14	Standard Class	BH-11710	Brosina Hoffman	Consumer	United States	Los Angeles	California	90032	West	FUR-FU-
7	7	CA-2018-115812	2018-06-09	2018-06-14	Standard Class	BH-11710	Brosina Hoffman	Consumer	United States	Los Angeles	California	90032	West	OFF-AR-
8	8	CA-2018-115812	2018-06-09	2018-06-14	Standard Class	BH-11710	Brosina Hoffman	Consumer	United States	Los Angeles	California	90032	West	TEC-PH-
9	9	CA-2018-115812	2018-06-09	2018-06-14	Standard Class	BH-11710	Brosina Hoffman	Consumer	United States	Los Angeles	California	90032	West	OFF-BI-1
10	10	CA-2018-115812	2018-06-09	2018-06-14	Standard Class	BH-11710	Brosina Hoffman	Consumer	United States	Los Angeles	California	90032	West	OFF-AP-

Query executed successfully. | DESKTOP-9HF98IA\SQLEXPRESS ... | DESKTOP-9HF98IA\Sunil ... | amazon | 00:00:00 | 9,815 rows

```sql
--5.)write a query to get all the orders where profit is negative

select * from orders where profit <0;
```

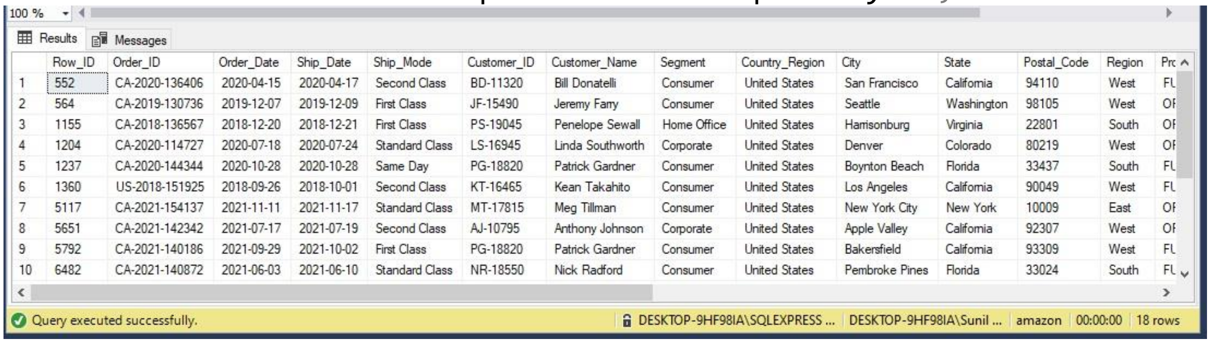


Results | Messages

	Row_ID	Order_ID	Order_Date	Ship_Date	Ship_Mode	Customer_ID	Customer_Name	Segment	Country_Region	City	State	Postal_Code	Region	Pr
1	4	US-2019-108966	2019-10-11	2019-10-18	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	Florida	33311	South	FU
2	15	US-2019-118983	2019-11-22	2019-11-26	Standard Class	HP-14815	Harold Pawlan	Home Office	United States	Fort Worth	Texas	76106	Central	OI
3	16	US-2019-118983	2019-11-22	2019-11-26	Standard Class	HP-14815	Harold Pawlan	Home Office	United States	Fort Worth	Texas	76106	Central	OI
4	24	US-2021-156909	2021-07-16	2021-07-18	Second Class	SF-20065	Sandra Flanagan	Consumer	United States	Philadelphia	Pennsylvania	19140	East	FU
5	28	US-2019-150630	2019-09-17	2019-09-21	Standard Class	TB-21520	Tracy Blumstein	Consumer	United States	Philadelphia	Pennsylvania	19140	East	FU
6	29	US-2019-150630	2019-09-17	2019-09-21	Standard Class	TB-21520	Tracy Blumstein	Consumer	United States	Philadelphia	Pennsylvania	19140	East	OI
7	33	US-2019-150630	2019-09-17	2019-09-21	Standard Class	TB-21520	Tracy Blumstein	Consumer	United States	Philadelphia	Pennsylvania	19140	East	OI
8	37	CA-2020-117590	2020-12-08	2020-12-10	First Class	GH-14485	Gene Hale	Corporate	United States	Richardson	Texas	75080	Central	FU
9	39	CA-2019-117415	2019-12-27	2019-12-31	Standard Class	SN-20710	Steve Nguyen	Home Office	United States	Houston	Texas	77041	Central	FU
10	40	CA-2019-117415	2019-12-27	2019-12-31	Standard Class	SN-20710	Steve Nguyen	Home Office	United States	Houston	Texas	77041	Central	FU

Query executed successfully. | DESKTOP-9HF98IA\SQLEXPRESS ... | DESKTOP-9HF98IA\Sunil ... | amazon | 00:00:00 | 1,871 rows

```
--6.)write a query to get all the orders where either quantity is
less than 3 or profit is 0

select * from orders where profit = 0 and quantity <3;
```



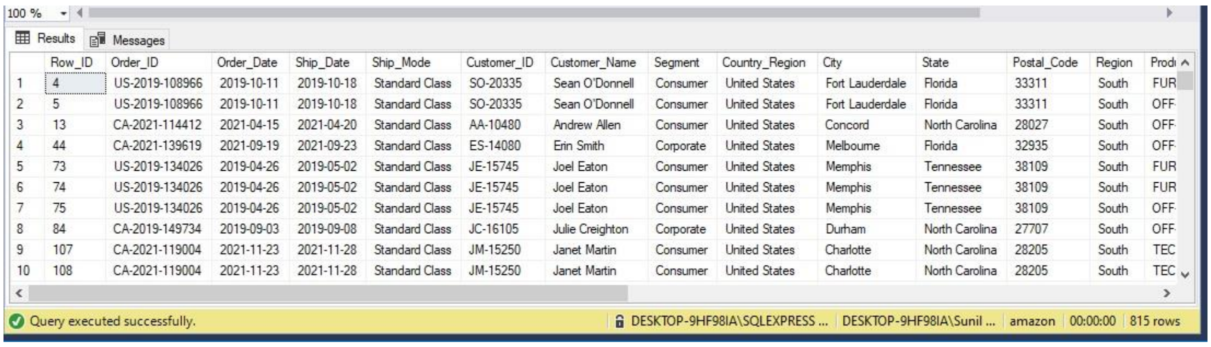
100 %

Results | Messages

	Row_ID	Order_ID	Order_Date	Ship_Date	Ship_Mode	Customer_ID	Customer_Name	Segment	Country_Region	City	State	Postal_Code	Region	Prc
1	552	CA-2020-136406	2020-04-15	2020-04-17	Second Class	BD-11320	Bill Donatelli	Consumer	United States	San Francisco	California	94110	West	FU
2	564	CA-2019-130736	2019-12-07	2019-12-09	First Class	JF-15490	Jeremy Farry	Consumer	United States	Seattle	Washington	98105	West	OF
3	1155	CA-2018-136567	2018-12-20	2018-12-21	First Class	PS-19045	Penelope Sewall	Home Office	United States	Harrisonburg	Virginia	22801	South	OF
4	1204	CA-2020-114727	2020-07-18	2020-07-24	Standard Class	LS-16945	Linda Southworth	Corporate	United States	Denver	Colorado	80219	West	OF
5	1237	CA-2020-144344	2020-10-28	2020-10-28	Same Day	PG-18820	Patrick Gardner	Consumer	United States	Boynton Beach	Florida	33437	South	FU
6	1360	US-2018-151925	2018-09-26	2018-10-01	Second Class	KT-16465	Kean Takahito	Consumer	United States	Los Angeles	California	90049	West	FU
7	5117	CA-2021-154137	2021-11-11	2021-11-17	Standard Class	MT-17815	Meg Tillman	Consumer	United States	New York City	New York	10009	East	OF
8	5651	CA-2021-142342	2021-07-17	2021-07-19	Second Class	AJ-10795	Anthony Johnson	Corporate	United States	Apple Valley	California	92307	West	OF
9	5792	CA-2021-140186	2021-09-29	2021-10-02	First Class	PG-18820	Patrick Gardner	Consumer	United States	Bakersfield	California	93309	West	FU
10	6482	CA-2021-140872	2021-06-03	2021-06-10	Standard Class	NR-18550	Nick Radford	Consumer	United States	Pembroke Pines	Florida	33024	South	FU

Query executed successfully. | DESKTOP-9HF98IA\SQLEXPRESS ... | DESKTOP-9HF98IA\Sunil ... | amazon | 00:00:00 | 18 rows

```
--7.) Your manager handles the sales for South region and he wants
you to create a report
--of all the orders in his region where some discount is provided to
the customers

select * from orders where region = 'South' and discount >0;
```



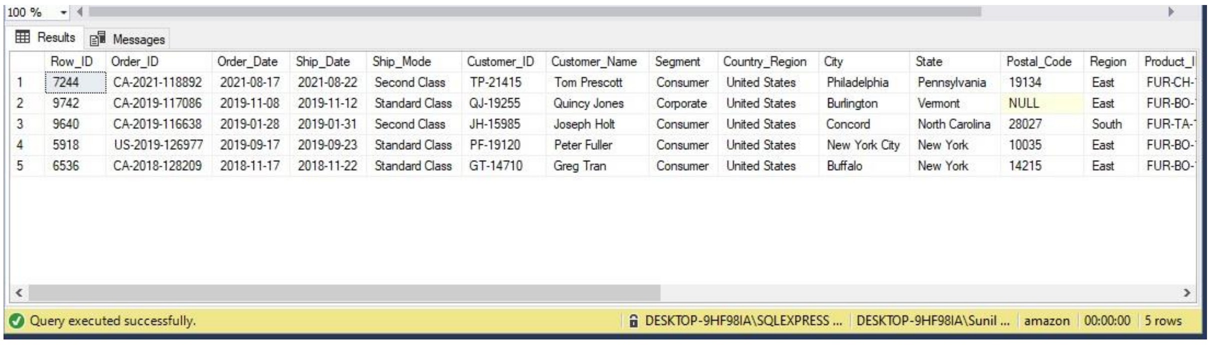
100 %

Results | Messages

	Row_ID	Order_ID	Order_Date	Ship_Date	Ship_Mode	Customer_ID	Customer_Name	Segment	Country_Region	City	State	Postal_Code	Region	Prodi
1	4	US-2019-108966	2019-10-11	2019-10-18	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	Florida	33311	South	FUR
2	5	US-2019-108966	2019-10-11	2019-10-18	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	Florida	33311	South	OFF
3	13	CA-2021-114412	2021-04-15	2021-04-20	Standard Class	AA-10480	Andrew Allen	Consumer	United States	Concord	North Carolina	28027	South	OFF
4	44	CA-2021-139619	2021-09-19	2021-09-23	Standard Class	ES-14080	Erin Smith	Corporate	United States	Melbourne	Florida	32935	South	OFF
5	73	US-2019-134026	2019-04-26	2019-05-02	Standard Class	JE-15745	Joel Eaton	Consumer	United States	Memphis	Tennessee	38109	South	FUR
6	74	US-2019-134026	2019-04-26	2019-05-02	Standard Class	JE-15745	Joel Eaton	Consumer	United States	Memphis	Tennessee	38109	South	FUR
7	75	US-2019-134026	2019-04-26	2019-05-02	Standard Class	JE-15745	Joel Eaton	Consumer	United States	Memphis	Tennessee	38109	South	OFF
8	84	CA-2019-149734	2019-09-03	2019-09-08	Standard Class	JC-16105	Julie Creighton	Corporate	United States	Durham	North Carolina	27707	South	OFF
9	107	CA-2021-119004	2021-11-23	2021-11-28	Standard Class	JM-15250	Janet Martin	Consumer	United States	Charlotte	North Carolina	28205	South	TEC
10	108	CA-2021-119004	2021-11-23	2021-11-28	Standard Class	JM-15250	Janet Martin	Consumer	United States	Charlotte	North Carolina	28205	South	TEC

Query executed successfully. | DESKTOP-9HF98IA\SQLEXPRESS ... | DESKTOP-9HF98IA\Sunil ... | amazon | 00:00:00 | 815 rows

```
--8.) Write a query to find top 5 orders with highest sales in
furniture category

select top 5 * from orders where category = 'Furniture' order by
sales DESC;
```



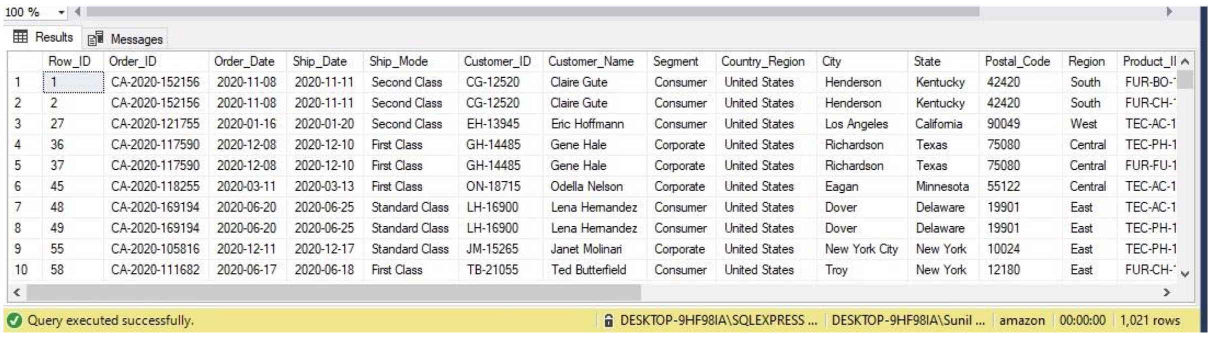
100 %

Results | Messages

	Row_ID	Order_ID	Order_Date	Ship_Date	Ship_Mode	Customer_ID	Customer_Name	Segment	Country_Region	City	State	Postal_Code	Region	Product_I
1	7244	CA-2021-118892	2021-08-17	2021-08-22	Second Class	TP-21415	Tom Prescott	Consumer	United States	Philadelphia	Pennsylvania	19134	East	FUR-CH-
2	9742	CA-2019-117086	2019-11-08	2019-11-12	Standard Class	QJ-19255	Quincy Jones	Corporate	United States	Burlington	Vermont	NULL	East	FUR-BO-
3	9640	CA-2019-116638	2019-01-28	2019-01-31	Second Class	JH-15985	Joseph Holt	Consumer	United States	Concord	North Carolina	28027	South	FUR-TA-
4	5918	US-2019-126977	2019-09-17	2019-09-23	Standard Class	PF-19120	Peter Fuller	Consumer	United States	New York City	New York	10035	East	FUR-BO-
5	6536	CA-2018-128209	2018-11-17	2018-11-22	Standard Class	GT-14710	Greg Tran	Consumer	United States	Buffalo	New York	14215	East	FUR-BO-

Query executed successfully. | DESKTOP-9HF98IA\SQLEXPRESS ... | DESKTOP-9HF98IA\Sunil ... | amazon | 00:00:00 | 5 rows

```sql
--9.)write a query to find all the records in technology and
furniture category
--for the orders placed in the year 2020 only

select * from orders where category in ('Technology','Furniture')
and
Order_date between '2020-01-01' and '2020-12-31';
```



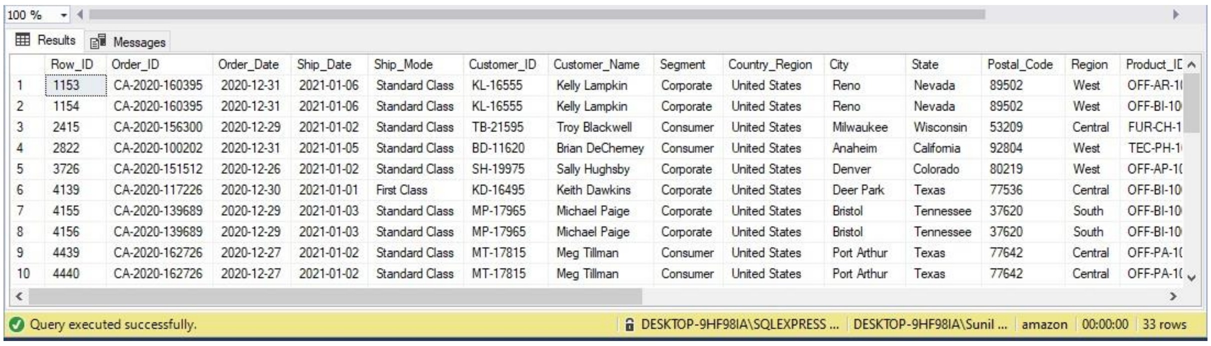
100 %

Results | Messages

	Row_ID	Order_ID	Order_Date	Ship_Date	Ship_Mode	Customer_ID	Customer_Name	Segment	Country_Region	City	State	Postal_Code	Region	Product_I
1	1	CA-2020-152156	2020-11-08	2020-11-11	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	Kentucky	42420	South	FUR-BO-
2	2	CA-2020-152156	2020-11-08	2020-11-11	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	Kentucky	42420	South	FUR-CH-
3	27	CA-2020-121755	2020-01-16	2020-01-20	Second Class	EH-13945	Eric Hoffmann	Consumer	United States	Los Angeles	California	90049	West	TEC-AC-1
4	36	CA-2020-117590	2020-12-08	2020-12-10	First Class	GH-14485	Gene Hale	Corporate	United States	Richardson	Texas	75080	Central	TEC-PH-1
5	37	CA-2020-117590	2020-12-08	2020-12-10	First Class	GH-14485	Gene Hale	Corporate	United States	Richardson	Texas	75080	Central	FUR-FU-1
6	45	CA-2020-118255	2020-03-11	2020-03-13	First Class	ON-18715	Odella Nelson	Corporate	United States	Eagan	Minnesota	55122	Central	TEC-AC-1
7	48	CA-2020-169194	2020-06-20	2020-06-25	Standard Class	LH-16900	Lena Hernandez	Consumer	United States	Dover	Delaware	19901	East	TEC-AC-1
8	49	CA-2020-169194	2020-06-20	2020-06-25	Standard Class	LH-16900	Lena Hernandez	Consumer	United States	Dover	Delaware	19901	East	TEC-PH-1
9	55	CA-2020-105816	2020-12-11	2020-12-17	Standard Class	JM-15265	Janet Molinari	Corporate	United States	New York City	New York	10024	East	TEC-PH-1
10	58	CA-2020-111682	2020-06-17	2020-06-18	First Class	TB-21055	Ted Butterfield	Consumer	United States	Troy	New York	12180	East	FUR-CH-

Query executed successfully. | DESKTOP-9HF98IA\SQLEXPRESS ... | DESKTOP-9HF98IA\Sunil ... | amazon | 00:00:00 | 1,021 rows

```sql
--10.)write a query to find all the orders where order date is in
year 2020 but ship date is in 2021

select * from orders where Order_Date between
'2020-01-01' and '2020-12-31' and ship_date between '2021-01-01' and
'2021-12-31';
```



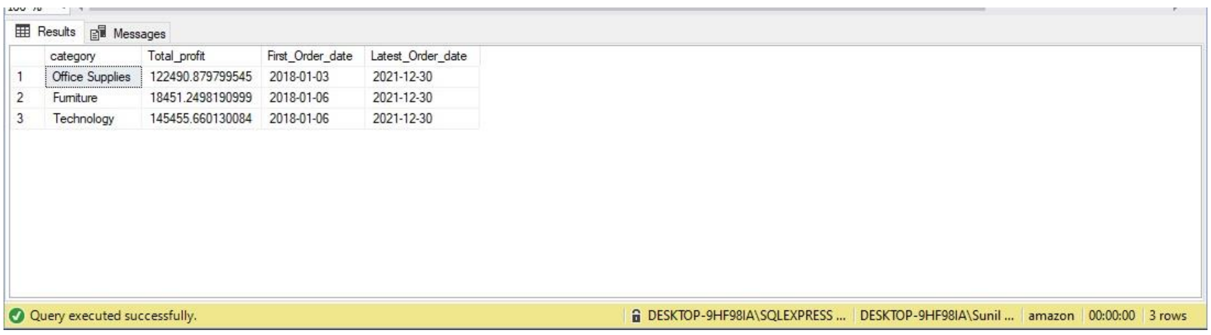
100 %

Results | Messages

	Row_ID	Order_ID	Order_Date	Ship_Date	Ship_Mode	Customer_ID	Customer_Name	Segment	Country_Region	City	State	Postal_Code	Region	Product_ID
1	1153	CA-2020-160395	2020-12-31	2021-01-06	Standard Class	KL-16555	Kelly Lampkin	Corporate	United States	Reno	Nevada	89502	West	OFF-AR-1
2	1154	CA-2020-160395	2020-12-31	2021-01-06	Standard Class	KL-16555	Kelly Lampkin	Corporate	United States	Reno	Nevada	89502	West	OFF-BI-10
3	2415	CA-2020-156300	2020-12-29	2021-01-02	Standard Class	TB-21595	Troy Blackwell	Consumer	United States	Milwaukee	Wisconsin	53209	Central	FUR-CH-1
4	2822	CA-2020-100202	2020-12-31	2021-01-05	Standard Class	BD-11620	Brian DeChemey	Consumer	United States	Anaheim	California	92804	West	TEC-PH-1
5	3726	CA-2020-151512	2020-12-26	2021-01-02	Standard Class	SH-19975	Sally Hughsby	Corporate	United States	Denver	Colorado	80219	West	OFF-AP-1
6	4139	CA-2020-117226	2020-12-30	2021-01-01	First Class	KD-16495	Keith Dawkins	Corporate	United States	Deer Park	Texas	77536	Central	OFF-BI-10
7	4155	CA-2020-139689	2020-12-29	2021-01-03	Standard Class	MP-17965	Michael Paige	Corporate	United States	Bristol	Tennessee	37620	South	OFF-BI-10
8	4156	CA-2020-139689	2020-12-29	2021-01-03	Standard Class	MP-17965	Michael Paige	Corporate	United States	Bristol	Tennessee	37620	South	OFF-BI-10
9	4439	CA-2020-162726	2020-12-27	2021-01-02	Standard Class	MT-17815	Meg Tillman	Consumer	United States	Port Arthur	Texas	77642	Central	OFF-PA-1
10	4440	CA-2020-162726	2020-12-27	2021-01-02	Standard Class	MT-17815	Meg Tillman	Consumer	United States	Port Arthur	Texas	77642	Central	OFF-PA-1

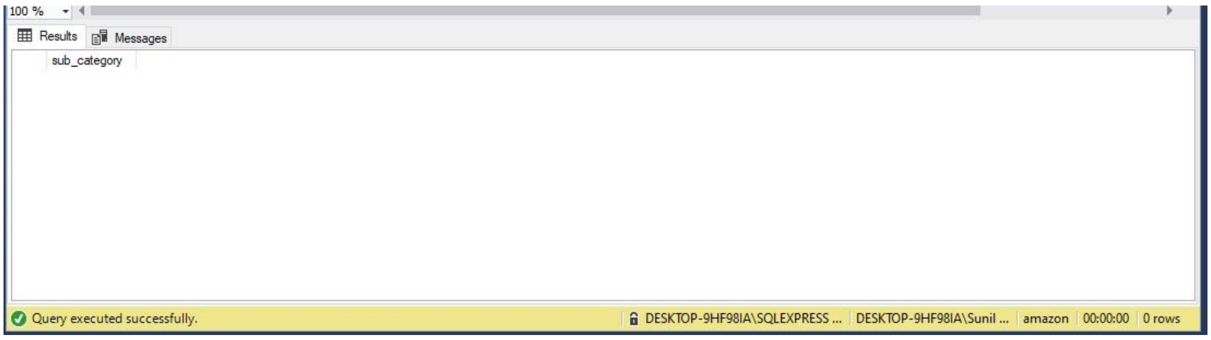
Query executed successfully. | DESKTOP-9HF98IA\SQLEXPRESS ... | DESKTOP-9HF98IA\Sunil ... | amazon | 00:00:00 | 33 rows

```sql
--11.)write a query to get total profit, first order date and latest
order date for each category

select category, sum(profit) as Total_profit, min(Order_Date) as
First_Order_date,
max(Order_Date) as Latest_Order_date from orders group by category;
```



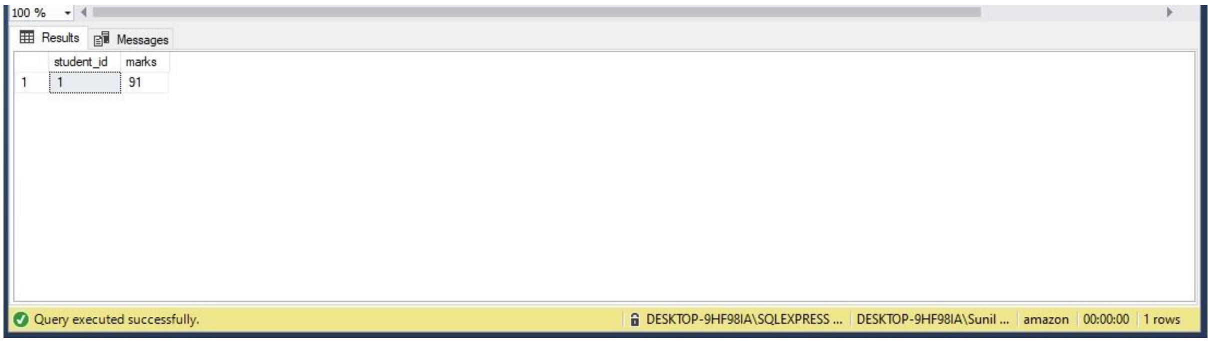
```
Results   Messages
   category         Total_profit       First_Order_date   Latest_Order_date
1  Office Supplies  122490.879799545   2018-01-03         2021-12-30
2  Furniture        18451.2498190999   2018-01-06         2021-12-30
3  Technology       145455.660130084   2018-01-06         2021-12-30

Query executed successfully.   DESKTOP-9HF98IA\SQLEXPRESS ...   DESKTOP-9HF98IA\Sunil ...   amazon   00:00:00   3 rows
```

```sql
--12.)write a query to find sub-categories where average profit is more than the half
--of the max profit in that sub-category

select sub_category from orders group by sub_category having
avg(profit) > max(profit)/2;
```



```
100 %
Results   Messages
   sub_category

Query executed successfully.   DESKTOP-9HF98IA\SQLEXPRESS ...   DESKTOP-9HF98IA\Sunil ...   amazon   00:00:00   0 rows
```

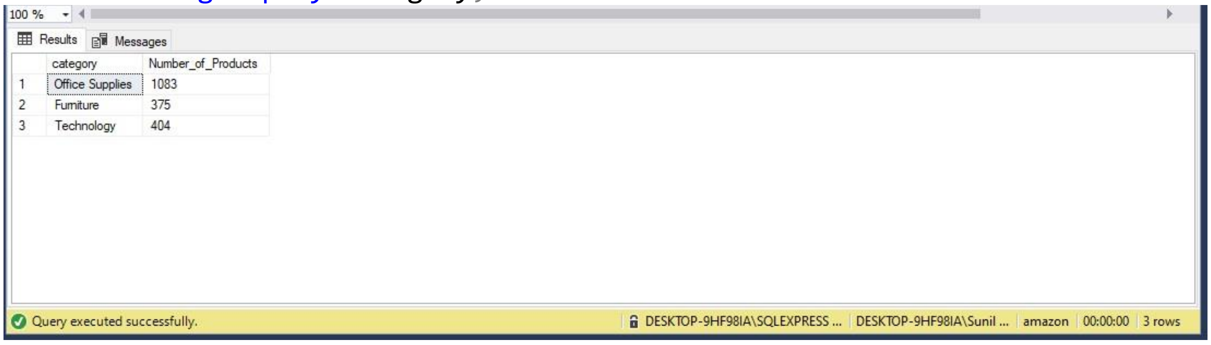
```sql
--13.)create the exams table with below script;

create table exams (student_id int, subject varchar(20), marks int);

insert into exams values(1,'Chemistry',91),(1,'Physics',91),(1,'Maths',92)
,(2,'Chemistry',80),(2,'Physics',90)
,(3,'Chemistry',80),(3,'Maths',80)
,(4,'Chemistry',71),(4,'Physics',54)
,(5,'Chemistry',79);


--14.)Write a query to find students who have got same marks in Physics
and Chemistry.

select student_id,marks from exams where subject in
('Physics','Chemistry')
group by student_id, marks having count(1)=2;
```



	student_id	marks
1	1	91

Query executed successfully. DESKTOP-9HF98IA\SQLEXPRESS ... DESKTOP-9HF98IA\Sunil ... amazon 00:00:00 1 rows

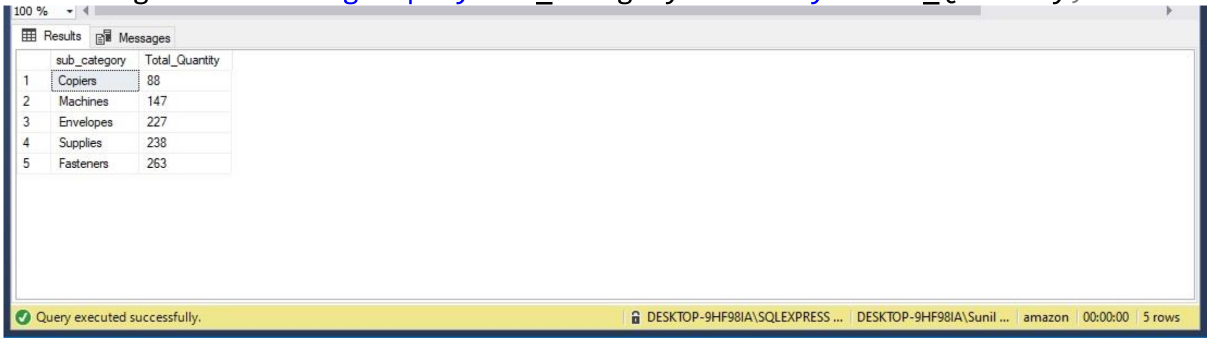
```
--15.)Write a query to find total number of products in each category.

select category, count(distinct product_id) as Number_of_Products
from orders group by category;
```



	category	Number_of_Products
1	Office Supplies	1083
2	Furniture	375
3	Technology	404

Query executed successfully. DESKTOP-9HF98IA\SQLEXPRESS ... DESKTOP-9HF98IA\Sunil ... amazon 00:00:00 3 rows

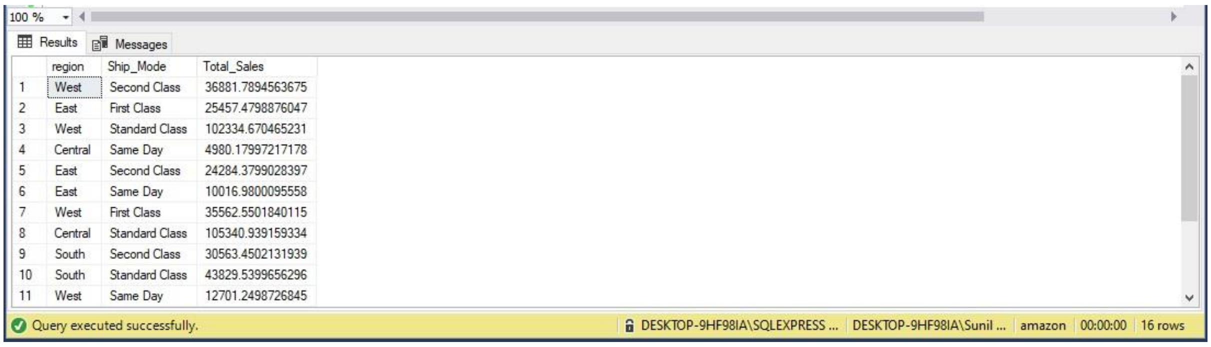
```
--16.)Write a query to find top 5 sub categories in west region by total
quantity sold

select top 5 sub_category, sum(quantity) as Total_Quantity from orders
where region ='West' group by sub_category order by Total_Quantity;
```



	sub_category	Total_Quantity
1	Copiers	88
2	Machines	147
3	Envelopes	227
4	Supplies	238
5	Fasteners	263

Query executed successfully. DESKTOP-9HF98IA\SQLEXPRESS ... DESKTOP-9HF98IA\Sunil ... amazon 00:00:00 5 rows

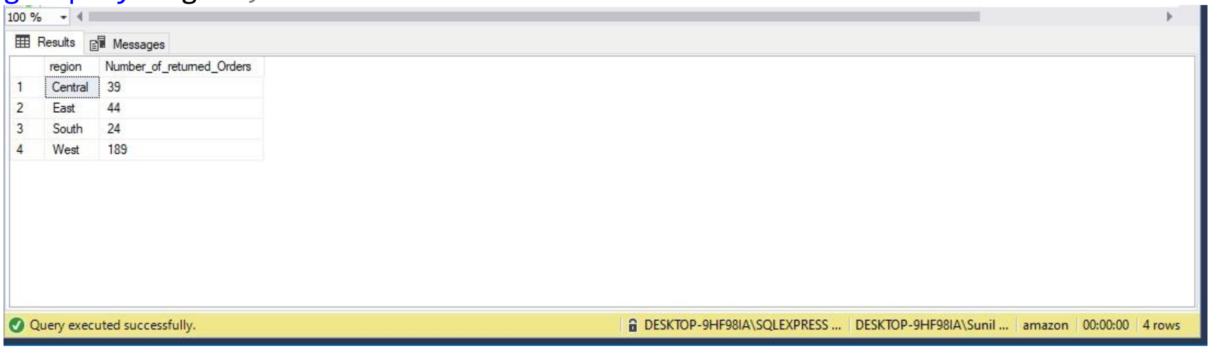
```
--17.) write a query to find total sales for each region and ship mode
combination for orders in year 2020

select region, Ship_Mode,sum(sales)as Total_Sales from orders
where order_date between '2020-01-01' and '2020-12-31'
group by region,Ship_Mode;
```



	region	Ship_Mode	Total_Sales
1	West	Second Class	36881.7894563675
2	East	First Class	25457.4798876047
3	West	Standard Class	102334.670465231
4	Central	Same Day	4980.17997217178
5	East	Second Class	24284.3799028397
6	East	Same Day	10016.9800095558
7	West	First Class	35562.5501840115
8	Central	Standard Class	105340.939159334
9	South	Second Class	30563.4502131939
10	South	Standard Class	43829.5399656296
11	West	Same Day	12701.2498726845

```
Query executed successfully.    DESKTOP-9HF98IA\SQLEXPRESS ...  DESKTOP-9HF98IA\Sunil ...  amazon  00:00:00  16 rows
```

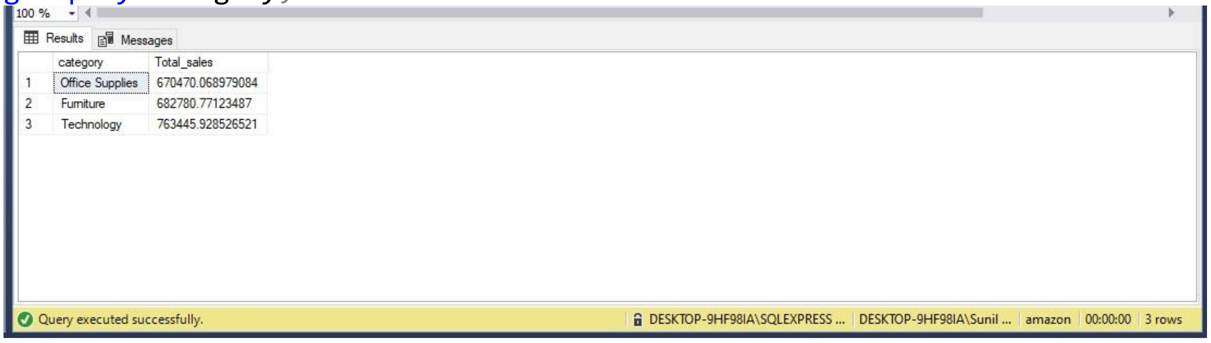
```
--18.)Write a query to get region wise count of return orders

select region,count(distinct o.order_id) as Number_of_returned_Orders
from orders o inner join returns r on o.order_id=r.order_id
group by region;
```



	region	Number_of_returned_Orders
1	Central	39
2	East	44
3	South	24
4	West	189

```
Query executed successfully.    DESKTOP-9HF98IA\SQLEXPRESS ...  DESKTOP-9HF98IA\Sunil ...  amazon  00:00:00  4 rows
```

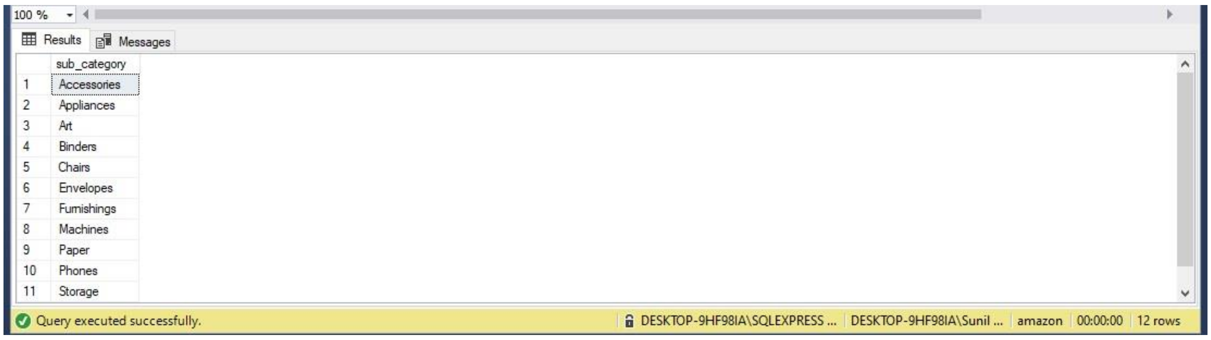
```
--19.)Write a query to get category wise sales of orders that were not
returned

select category, sum(o.sales) as Total_sales from orders o left join
returns r on o.order_id = r.order_id where r.order_id is null
group by category;
```



	category	Total_sales
1	Office Supplies	670470.068979084
2	Furniture	682780.77123487
3	Technology	763445.928526521

```
Query executed successfully.    DESKTOP-9HF98IA\SQLEXPRESS ...  DESKTOP-9HF98IA\Sunil ...  amazon  00:00:00  3 rows
```

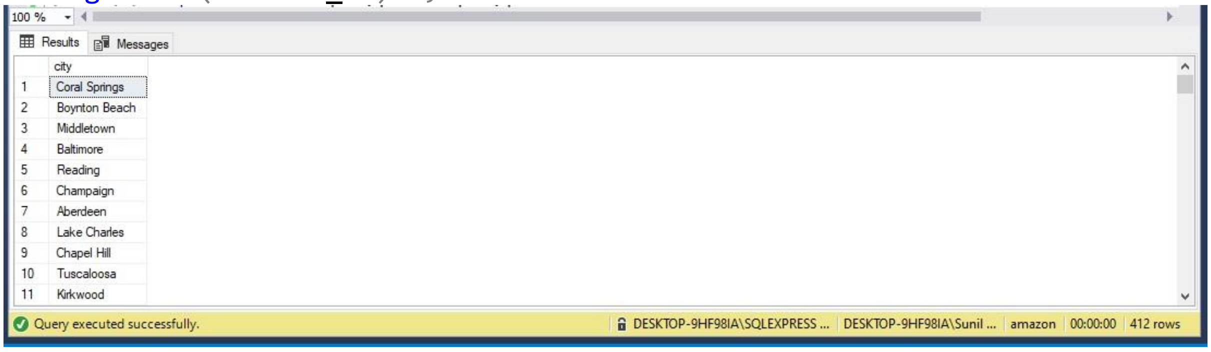
```
--20.)Write a query to print sub categories where we have all 3 kinds of
returns (others,bad quality,wrong items)

select o.sub_category from orders o inner join returns r
on o.order_id = r.order_id group by o.sub_category
having count(distinct r.return_reason)=3;
```


100 % Results Messages

	sub_category
1	Accessories
2	Appliances
3	Art
4	Binders
5	Chairs
6	Envelopes
7	Furnishings
8	Machines
9	Paper
10	Phones
11	Storage

Query executed successfully. | DESKTOP-9HF98IA\SQLEXPRESS ... | DESKTOP-9HF98IA\Sunil ... | amazon | 00:00:00 | 12 rows

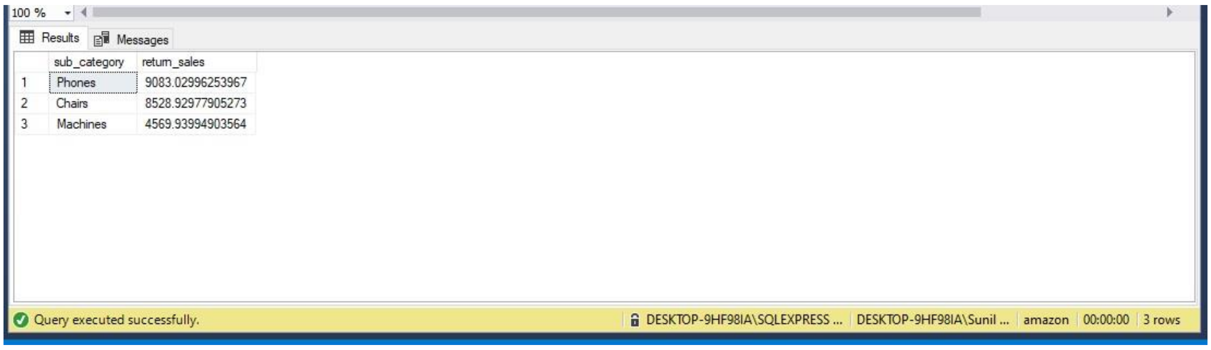
```
--21.)Write a query to find cities where not even a single order was
returned.

select o.city from orders o left join returns r
on o.order_id=r.order_id
group by city
having count(r.order_id)=0;
```


100 % Results Messages

	city
1	Coral Springs
2	Boynton Beach
3	Middletown
4	Baltimore
5	Reading
6	Champaign
7	Aberdeen
8	Lake Charles
9	Chapel Hill
10	Tuscaloosa
11	Kirkwood

Query executed successfully. | DESKTOP-9HF98IA\SQLEXPRESS ... | DESKTOP-9HF98IA\Sunil ... | amazon | 00:00:00 | 412 rows

```
--22.)Write a query to find top 3 subcategories by sales of returned
orders in east region

select top 3 sub_category,sum(o.sales) as return_sales
from orders o
inner join returns r on o.order_id=r.order_id
where o.region='East'
group by sub_category
order by return_sales  desc;
```

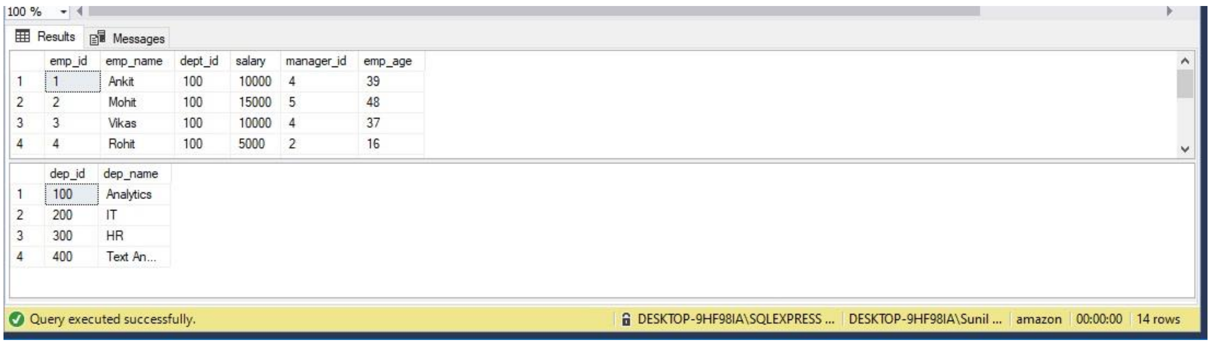

100 % Results Messages

	sub_category	return_sales
1	Phones	9083.02996253967
2	Chairs	8528.92977905273
3	Machines	4569.93994903564

Query executed successfully. | DESKTOP-9HF98IA\SQLEXPRESS ... | DESKTOP-9HF98IA\Sunil ... | amazon | 00:00:00 | 3 rows

```
create table employee(
    emp_id int,
    emp_name varchar(20),
    dept_id int,
    salary int,
    manager_id int,
```

```sql
    emp_age int
);


insert into employee values(1,'Ankit',100,10000,4,39);
insert into employee values(2,'Mohit',100,15000,5,48);
insert into employee values(3,'Vikas',100,10000,4,37);
insert into employee values(4,'Rohit',100,5000,2,16);
insert into employee values(5,'Mudit',200,12000,6,55);
insert into employee values(6,'Agam',200,12000,2,14);
insert into employee values(7,'Sanjay',200,9000,2,13);
insert into employee values(8,'Ashish',200,5000,2,12);
insert into employee values(9,'Mukesh',300,6000,6,51);
insert into employee values(10,'Rakesh',500,7000,6,50);
select * from employee;

create table dept(
    dep_id int,
    dep_name varchar(20)
);
insert into dept values(100,'Analytics');
insert into dept values(200,'IT');
insert into dept values(300,'HR');
insert into dept values(400,'Text Analytics');
select * from dept;

select * from employee;
select * from dept;
```

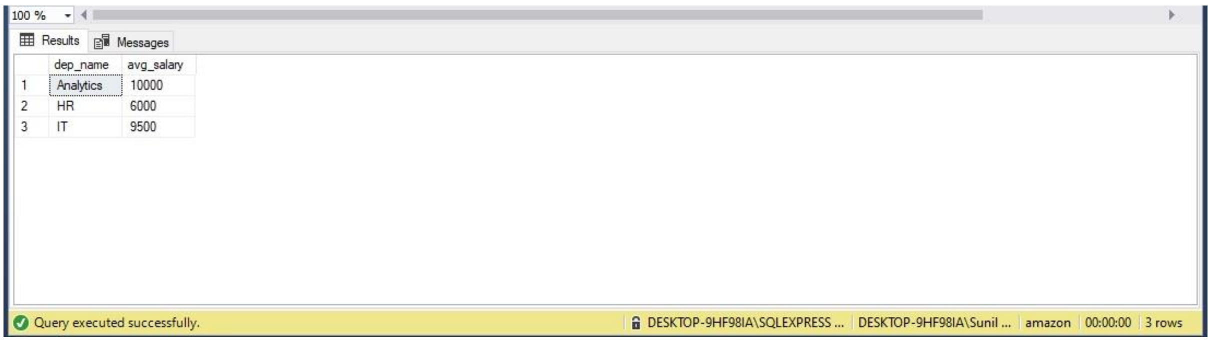


	emp_id	emp_name	dept_id	salary	manager_id	emp_age
1	1	Ankit	100	10000	4	39
2	2	Mohit	100	15000	5	48
3	3	Vikas	100	10000	4	37
4	4	Rohit	100	5000	2	16

	dep_id	dep_name
1	100	Analytics
2	200	IT
3	300	HR
4	400	Text An...

```
Query executed successfully.    DESKTOP-9HF98IA\SQLEXPRESS ...  DESKTOP-9HF98IA\Sunil ...  amazon  00:00:00  14 rows
```

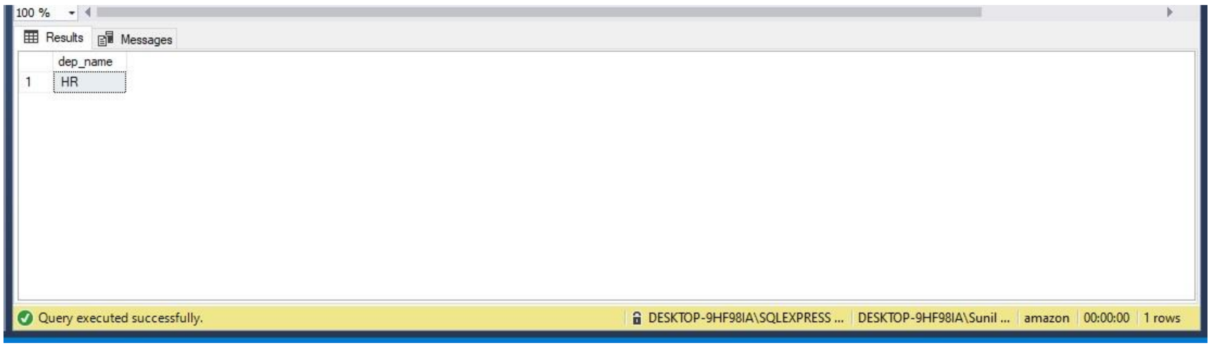
```sql
--23.)write a query to print dep name and average salary of employees in
that dep

select d.dep_name, avg(e.salary)as avg_salary from employee e inner join
dept d on
d.dep_id = e.dept_id group by d.dep_name;
```



	dep_name	avg_salary
1	Analytics	10000
2	HR	6000
3	IT	9500

```
Query executed successfully.    DESKTOP-9HF98IA\SQLEXPRESS ...  DESKTOP-9HF98IA\Sunil ...  amazon  00:00:00  3 rows
```

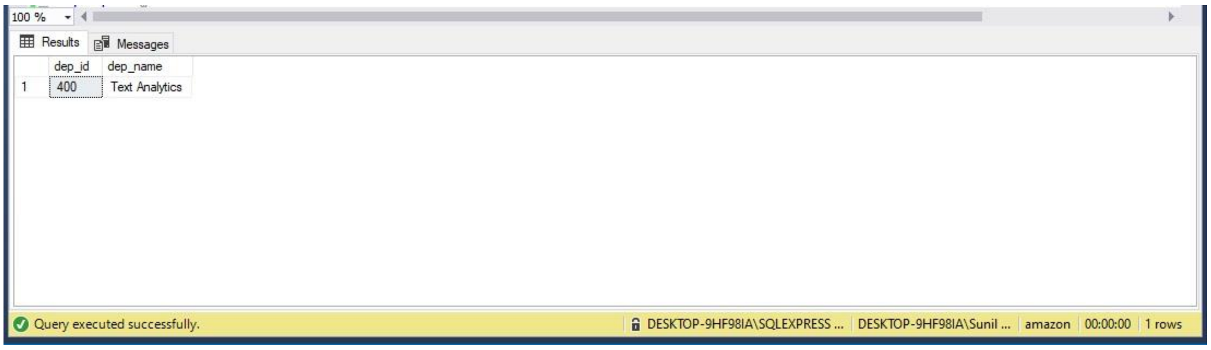
```
--24.)Write a query to print dep names where none of the emplyees have
same salary.

select d.dep_name from employee e join dept d
on e.dept_id = d.dep_id
group by d.dep_name
having count(e.emp_id)=count(distinct e.salary);
```



	dep_name
1	HR

```
Query executed successfully.    DESKTOP-9HF98IA\SQLEXPRESS ... | DESKTOP-9HF98IA\Sunil ... | amazon | 00:00:00 | 1 rows
```

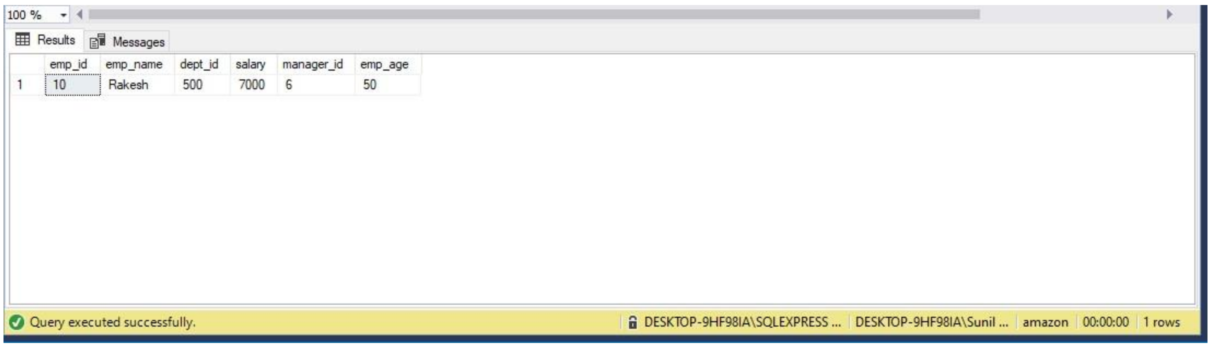
```
--25.)Write a query to print dep name for which there is no employee

select * from employee;
select * from dept;

select d.dep_id,d.dep_name
from dept d
left join employee e on e.dept_id=d.dep_id
group by d.dep_id,d.dep_name
having count(e.emp_id)=0;
```



	dep_id	dep_name
1	400	Text Analytics

```
Query executed successfully.    DESKTOP-9HF98IA\SQLEXPRESS ... | DESKTOP-9HF98IA\Sunil ... | amazon | 00:00:00 | 1 rows
```

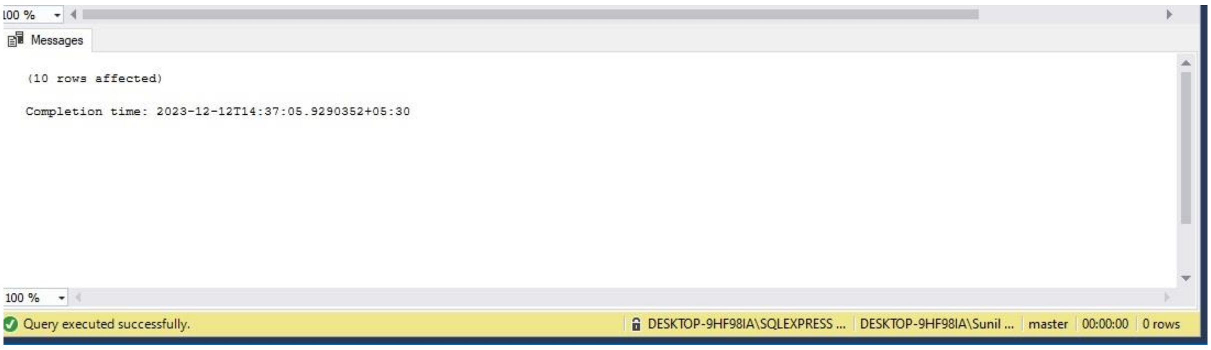
```
--26.)Write a query to print employees name for which dep id is not
avaiable in dept table

select e.*
from employee e
left join dept d  on e.dept_id=d.dep_id
where d.dep_id is null;
```



	emp_id	emp_name	dept_id	salary	manager_id	emp_age
1	10	Rakesh	500	7000	6	50

```
Query executed successfully.    DESKTOP-9HF98IA\SQLEXPRESS ... | DESKTOP-9HF98IA\Sunil ... | amazon | 00:00:00 | 1 rows
```

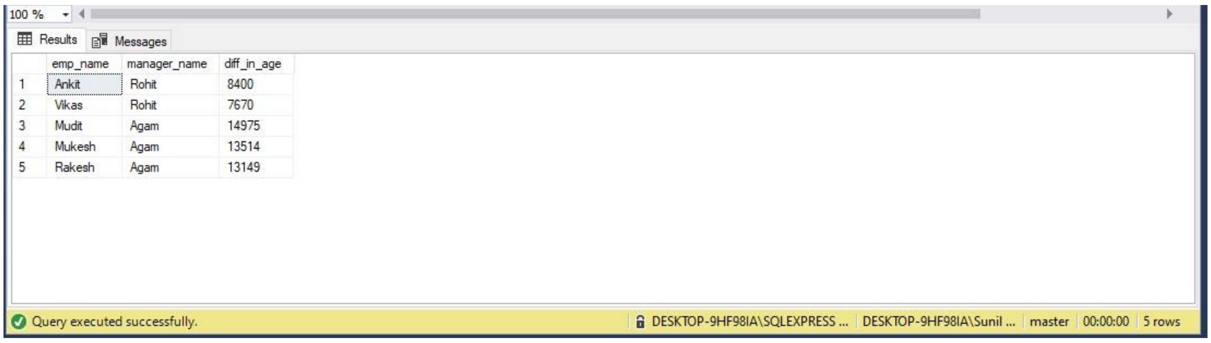
```
--27.)Run the following command to add and update dob column in employee table
alter table  employee add dob date;
update employee set dob = dateadd(year,-1*emp_age,getdate());
```



```
(10 rows affected)

Completion time: 2023-12-12T14:37:05.9290352+05:30
```

```
Query executed successfully.    DESKTOP-9HF98IA\SQLEXPRESS ... | DESKTOP-9HF98IA\Sunil ... | master | 00:00:00 | 0 rows
```

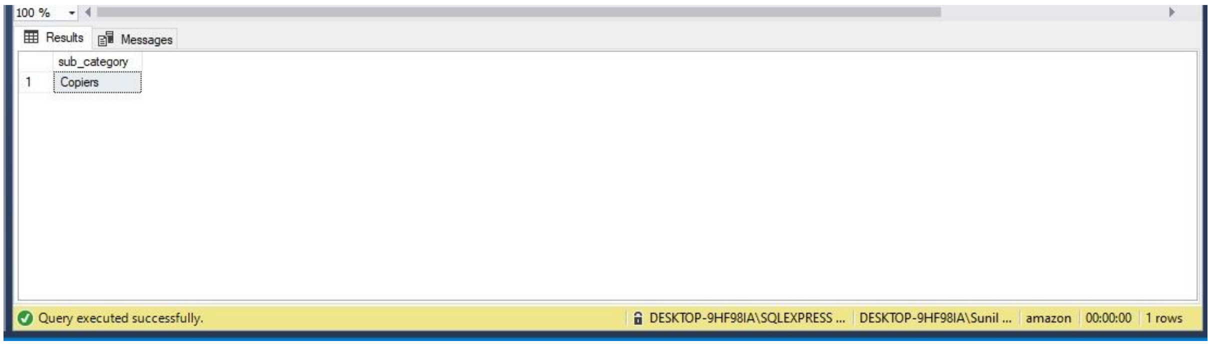
```
--28.) write a query to print emp name , their manager name and diffrence in their age
(in days)
--for employees whose year of birth is before their managers year of birth

select e1.emp_name,e2.emp_name as manager_name , DATEDIFF(day,e1.dob,e2.dob) as
diff_in_age
from employee e1
inner join employee e2 on e1.manager_id=e2.emp_id
where DATEPART(year,e1.dob)< DATEPART(year,e2.dob);
```



	emp_name	manager_name	diff_in_age
1	Ankit	Rohit	8400
2	Vikas	Rohit	7670
3	Mudit	Agam	14975
4	Mukesh	Agam	13514
5	Rakesh	Agam	13149

```
Query executed successfully.    DESKTOP-9HF98IA\SQLEXPRESS ... | DESKTOP-9HF98IA\Sunil ... | master | 00:00:00 | 5 rows
```

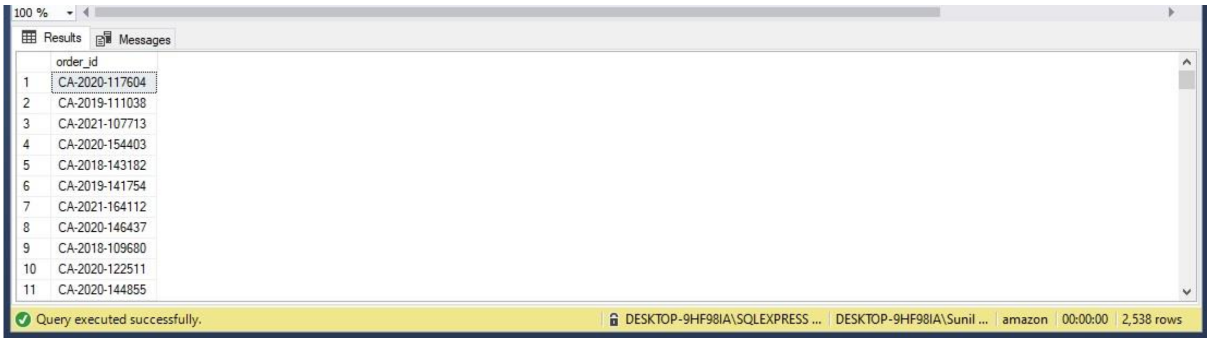
```
--29.)write a query to find subcategories who never had any return orders in the month
of november (irrespective of years)

select sub_category
from orders o
left join returns r on o.order_id=r.order_id
where DATEPART(month,order_date)=11
group by sub_category
having count(r.order_id)=0;
```



	sub_category
1	Copiers

Query executed successfully. | DESKTOP-9HF98IA\SQLEXPRESS ... | DESKTOP-9HF98IA\Sunil ... | amazon | 00:00:00 | 1 rows

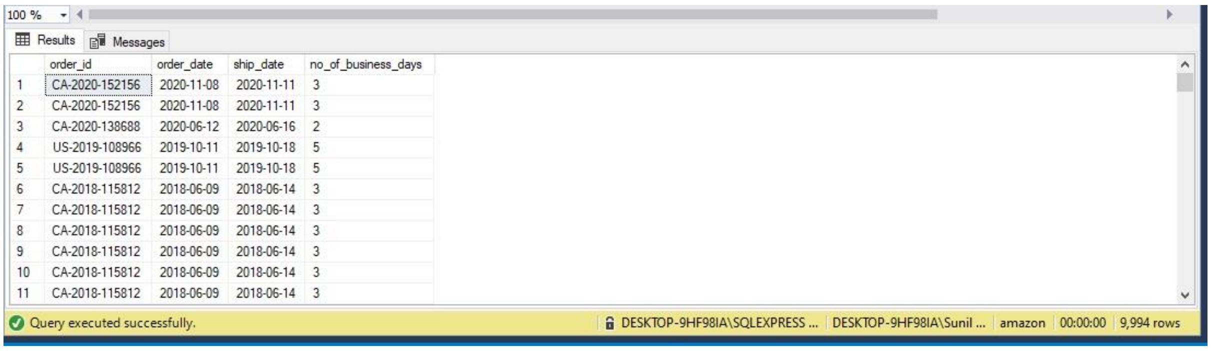
```
--30.)orders table can have multiple rows for a particular order_id when customers
buys more than 1 product in an order.
--write a query to find order ids where there is only 1 product bought by the
customer.

select order_id
from orders
group by order_id
having count(1)=1;
```



	order_id
1	CA-2020-117604
2	CA-2019-111038
3	CA-2021-107713
4	CA-2020-154403
5	CA-2018-143182
6	CA-2019-141754
7	CA-2021-164112
8	CA-2020-146437
9	CA-2018-109680
10	CA-2020-122511
11	CA-2020-144855

Query executed successfully. | DESKTOP-9HF98IA\SQLEXPRESS ... | DESKTOP-9HF98IA\Sunil ... | amazon | 00:00:00 | 2,538 rows

```
--31.)write a query to get number of business days between order_date and ship_date
(exclude weekends).
--Assume that all order date and ship date are on weekdays only

select order_id,order_date,ship_date ,datediff(day,order_date,ship_date)-
2*datediff(week,order_date,ship_date)
as no_of_business_days
from
orders;
```

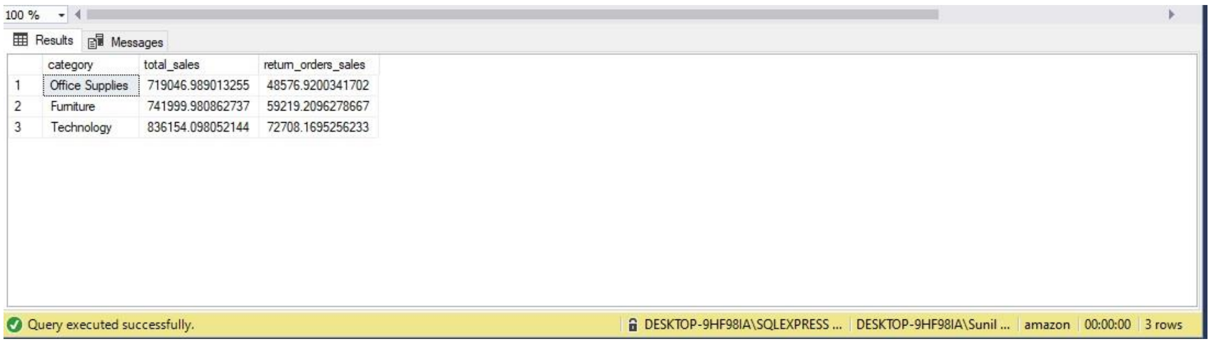


	order_id	order_date	ship_date	no_of_business_days
1	CA-2020-152156	2020-11-08	2020-11-11	3
2	CA-2020-152156	2020-11-08	2020-11-11	3
3	CA-2020-138688	2020-06-12	2020-06-16	2
4	US-2019-108966	2019-10-11	2019-10-18	5
5	US-2019-108966	2019-10-11	2019-10-18	5
6	CA-2018-115812	2018-06-09	2018-06-14	3
7	CA-2018-115812	2018-06-09	2018-06-14	3
8	CA-2018-115812	2018-06-09	2018-06-14	3
9	CA-2018-115812	2018-06-09	2018-06-14	3
10	CA-2018-115812	2018-06-09	2018-06-14	3
11	CA-2018-115812	2018-06-09	2018-06-14	3

Query executed successfully. | DESKTOP-9HF98IA\SQLEXPRESS ... | DESKTOP-9HF98IA\Sunil ... | amazon | 00:00:00 | 9,994 rows

```
--32.)write a query to print 3 columns : category, total_sales and (total sales of
returned orders)
```

```sql
select o.category,sum(o.sales) as total_sales
,sum(case when r.order_id is not null then sales end) as return_orders_sales
from orders o
left join returns r on o.order_id=r.order_id
group by category;
```

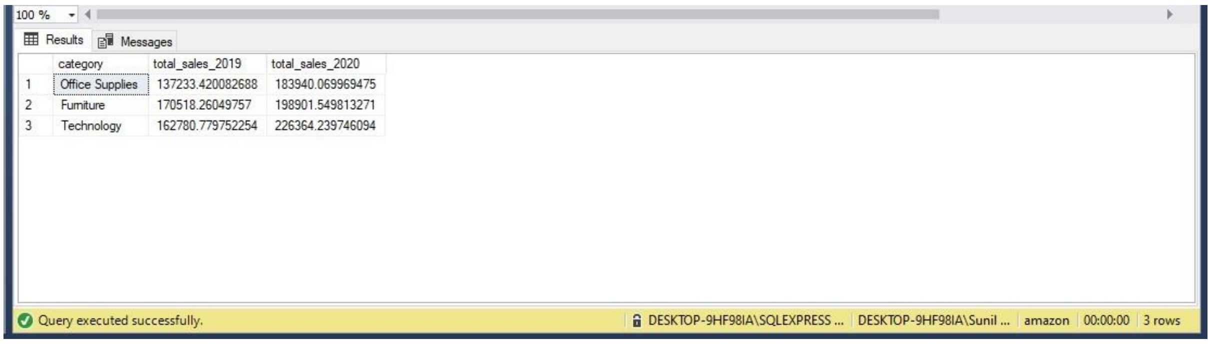


100 % | Results | Messages

	category	total_sales	return_orders_sales
1	Office Supplies	719046.989013255	48576.9200341702
2	Furniture	741999.980862737	59219.2096278667
3	Technology	836154.098052144	72708.1695256233

Query executed successfully. | DESKTOP-9HF98IA\SQLEXPRESS ... | DESKTOP-9HF98IA\Sunil ... | amazon | 00:00:00 | 3 rows

```sql
--33.)write a query to print below 3 columns
--category, total_sales_2019(sales in year 2019), total_sales_2020(sales in year 2020)

select category,sum(case when datepart(year,order_date)=2019 then sales end) as
total_sales_2019
,sum(case when datepart(year,order_date)=2020 then sales end) as total_sales_2020
from orders
group by category;
```

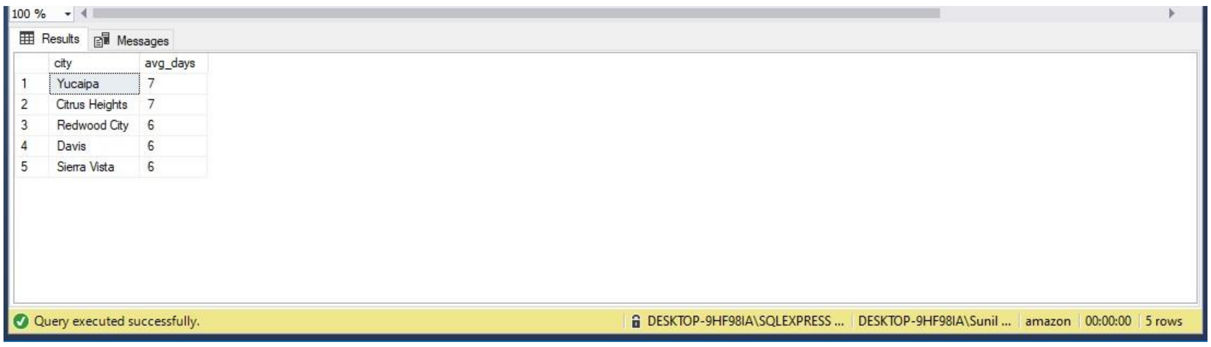


100 % | Results | Messages

	category	total_sales_2019	total_sales_2020
1	Office Supplies	137233.420082688	183940.069969475
2	Furniture	170518.26049757	198901.549813271
3	Technology	162780.779752254	226364.239746094

Query executed successfully. | DESKTOP-9HF98IA\SQLEXPRESS ... | DESKTOP-9HF98IA\Sunil ... | amazon | 00:00:00 | 3 rows

```sql
--34.)write a query print top 5 cities in west region by average no of days between
order date and ship date.

select top 5 city, avg(datediff(day,order_date,ship_date) ) as avg_days
from orders
where region='West'
group by city
order by avg_days desc;
```



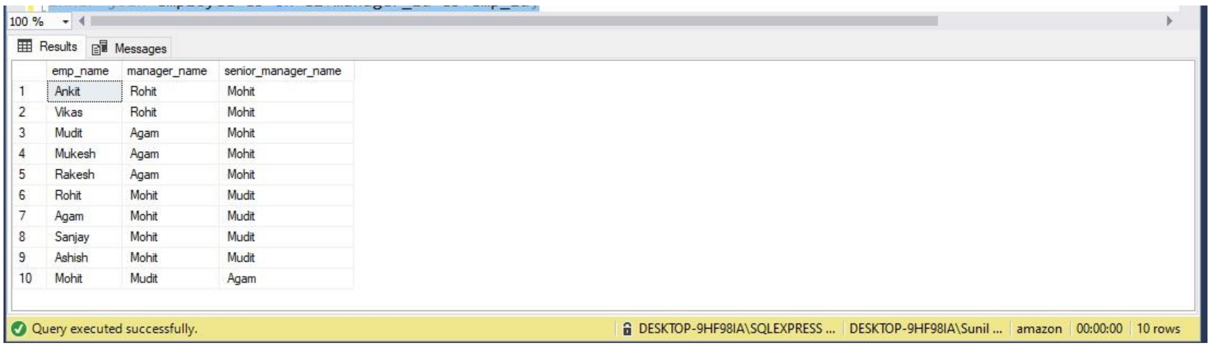
100 %

Results | Messages

	city	avg_days
1	Yucaipa	7
2	Citrus Heights	7
3	Redwood City	6
4	Davis	6
5	Sierra Vista	6

```
Query executed successfully.    DESKTOP-9HF98IA\SQLEXPRESS ... | DESKTOP-9HF98IA\Sunil ... | amazon | 00:00:00 | 5 rows
```

```
--35.)write a query to print emp name, manager name and senior manager name (senior
manager is manager's manager)

select e1.emp_name,e2.emp_name as manager_name,e3.emp_name as senior_manager_name
from employee e1
inner join employee e2 on e1.manager_id=e2.emp_id
inner join employee e3 on e2.manager_id=e3.emp_id;
```



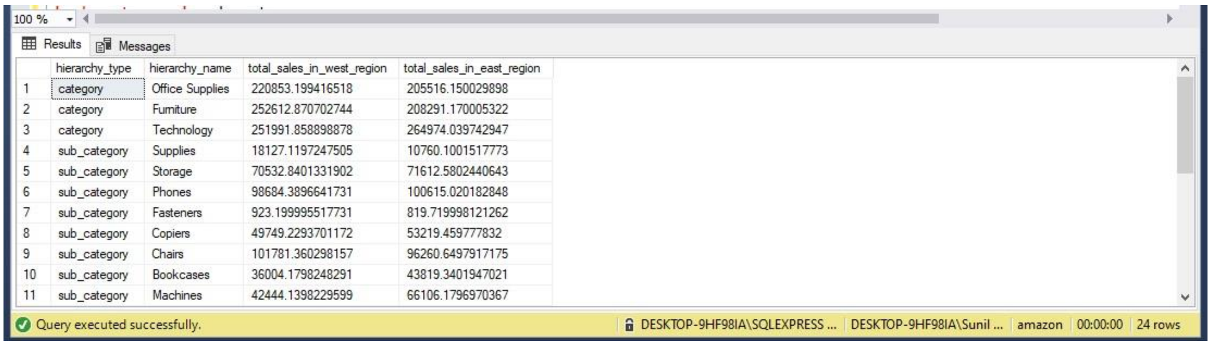
100 %

Results | Messages

	emp_name	manager_name	senior_manager_name
1	Ankit	Rohit	Mohit
2	Vikas	Rohit	Mohit
3	Mudit	Agam	Mohit
4	Mukesh	Agam	Mohit
5	Rakesh	Agam	Mohit
6	Rohit	Mohit	Mudit
7	Agam	Mohit	Mudit
8	Sanjay	Mohit	Mudit
9	Ashish	Mohit	Mudit
10	Mohit	Mudit	Agam

```
Query executed successfully.    DESKTOP-9HF98IA\SQLEXPRESS ... | DESKTOP-9HF98IA\Sunil ... | amazon | 00:00:00 | 10 rows
```

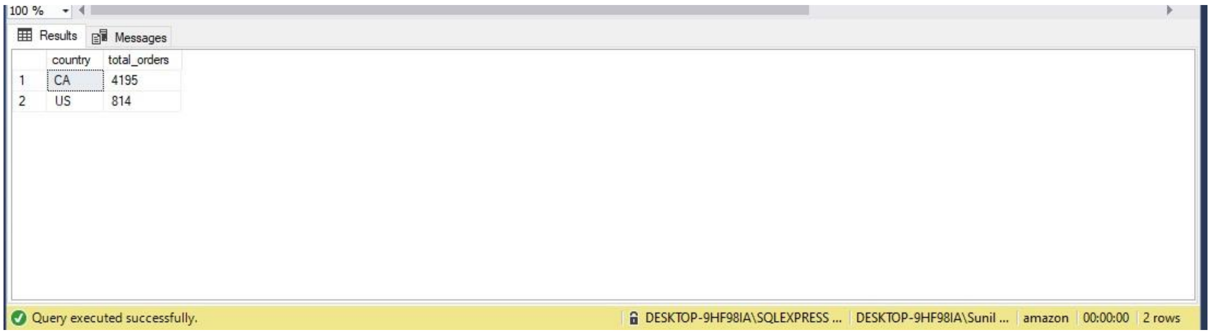
```
--36.)Write a query to print customer name and no of occurence of character 'n' in the
customer name.

select
'category' as hierarchy_type,category as hierarchy_name
,sum(case when region='West' then sales end) as total_sales_in_west_region
,sum(case when region='East' then sales end) as total_sales_in_east_region
from orders
group by category
union all
select
'sub_category',sub_category
,sum(case when region='West' then sales end) as total_sales_in_west_region
,sum(case when region='East' then sales end) as total_sales_in_east_region
from orders
group by sub_category
union all
select
'ship_mode ',ship_mode
,sum(case when region='West' then sales end) as total_sales_in_west_region
,sum(case when region='East' then sales end) as total_sales_in_east_region
from orders
group by ship_mode;
```



	hierarchy_type	hierarchy_name	total_sales_in_west_region	total_sales_in_east_region
1	category	Office Supplies	220853.199416518	205516.150029898
2	category	Fumiture	252612.870702744	208291.170005322
3	category	Technology	251991.858898878	264974.039742947
4	sub_category	Supplies	18127.1197247505	10760.1001517773
5	sub_category	Storage	70532.8401331902	71612.5802440643
6	sub_category	Phones	98684.3896641731	100615.020182848
7	sub_category	Fasteners	923.199995517731	819.719998121262
8	sub_category	Copiers	49749.2293701172	53219.459777832
9	sub_category	Chairs	101781.360298157	96260.6497917175
10	sub_category	Bookcases	36004.1798248291	43819.3401947021
11	sub_category	Machines	42444.1398229599	66106.1796970367

Query executed successfully. | DESKTOP-9HF98IA\SQLEXPRESS ... | DESKTOP-9HF98IA\Sunil ... | amazon | 00:00:00 | 24 rows

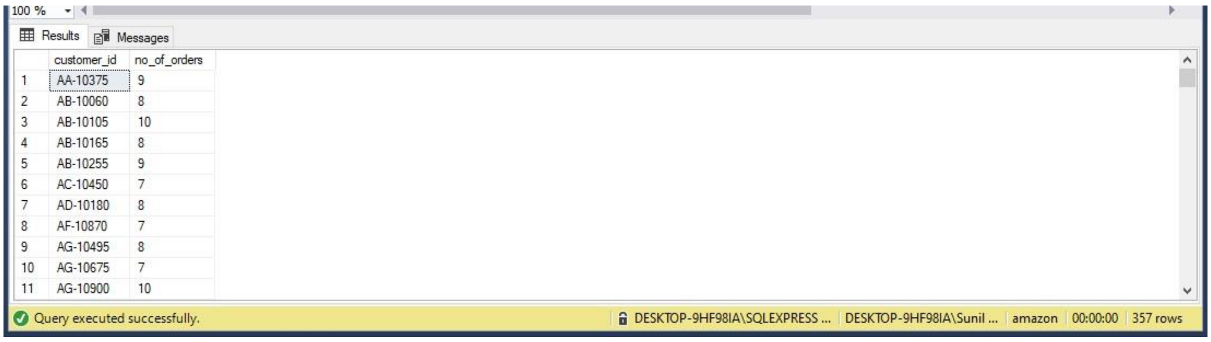
```
--37.)--the first 2 characters of order_id represents the country of order placed .
write a query to print total no of orders placed in each country
--(an order can have 2 rows in the data when more than 1 item was purchased in the
order but it should be considered as 1 order)

select left(order_id,2) as country, count(distinct order_id) as total_orders
from orders
group by left(order_id,2);
```



	country	total_orders
1	CA	4195
2	US	814

Query executed successfully. | DESKTOP-9HF98IA\SQLEXPRESS ... | DESKTOP-9HF98IA\Sunil ... | amazon | 00:00:00 | 2 rows

```
--38.)write a query to find premium customers from orders data.
--Premium customers are those who have done more orders than average no of orders per
customer.


with no_of_orders_each_customer as (
select customer_id,count(distinct order_id) as no_of_orders
from orders
group by customer_id)
select * from
no_of_orders_each_customer where no_of_orders > (select avg(no_of_orders) from
no_of_orders_each_customer);
```

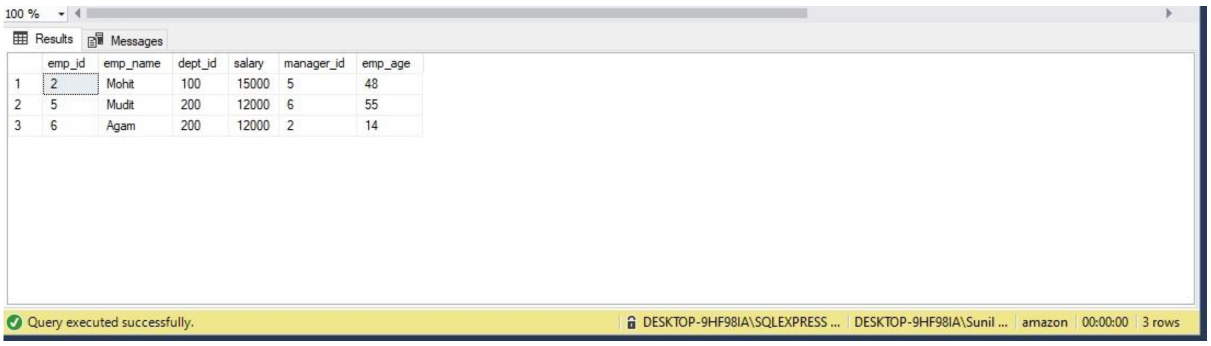


	customer_id	no_of_orders
1	AA-10375	9
2	AB-10060	8
3	AB-10105	10
4	AB-10165	8
5	AB-10255	9
6	AC-10450	7
7	AD-10180	8
8	AF-10870	7
9	AG-10495	8
10	AG-10675	7
11	AG-10900	10

Query executed successfully. | DESKTOP-9HF98IA\SQLEXPRESS ... | DESKTOP-9HF98IA\Sunil ... | amazon | 00:00:00 | 357 rows

```
--39.)write a query to find employees whose salary is more than average salary of
employees in their department.
```

```
select e.* from employee e
inner join (select dept_id,avg(salary) as avg_sal from employee group by dept_id)  d
on e.dept_id=d.dept_id
where salary>avg_sal;
```

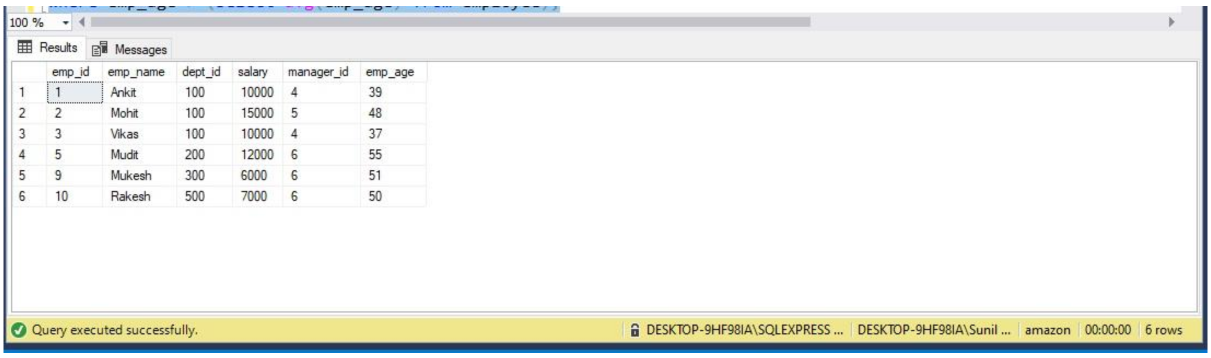


100 % | Results | Messages

	emp_id	emp_name	dept_id	salary	manager_id	emp_age
1	2	Mohit	100	15000	5	48
2	5	Mudit	200	12000	6	55
3	6	Agam	200	12000	2	14

Query executed successfully. | DESKTOP-9HF98IA\SQLEXPRESS ... | DESKTOP-9HF98IA\Sunil ... | amazon | 00:00:00 | 3 rows

```
--40.)write a query to find employees whose age is more than average age of all the
employees.

select * from employee
where emp_age > (select avg(emp_age) from employee);
```

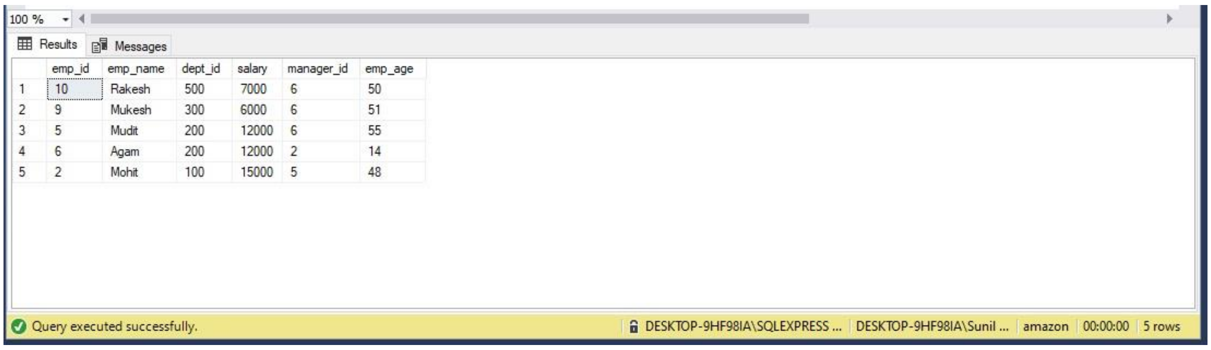


100 % | Results | Messages

	emp_id	emp_name	dept_id	salary	manager_id	emp_age
1	1	Ankit	100	10000	4	39
2	2	Mohit	100	15000	5	48
3	3	Vikas	100	10000	4	37
4	5	Mudit	200	12000	6	55
5	9	Mukesh	300	6000	6	51
6	10	Rakesh	500	7000	6	50

Query executed successfully. | DESKTOP-9HF98IA\SQLEXPRESS ... | DESKTOP-9HF98IA\Sunil ... | amazon | 00:00:00 | 6 rows

```
--41.)write a query to print emp name, salary and dep id of highest salaried employee
in each department.

select e.* from employee e
inner join (select dept_id,max(salary) as max_sal from employee group by dept_id)  d
on e.dept_id=d.dept_id
where salary=max_sal;
```

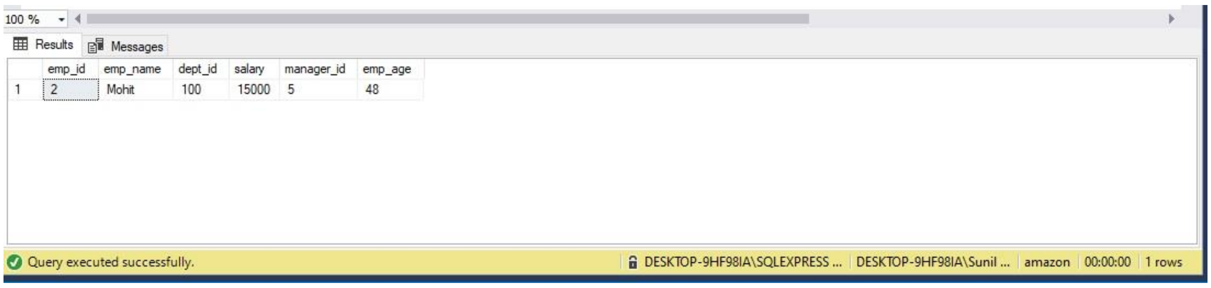


100 % | Results | Messages

	emp_id	emp_name	dept_id	salary	manager_id	emp_age
1	10	Rakesh	500	7000	6	50
2	9	Mukesh	300	6000	6	51
3	5	Mudit	200	12000	6	55
4	6	Agam	200	12000	2	14
5	2	Mohit	100	15000	5	48

Query executed successfully. | DESKTOP-9HF98IA\SQLEXPRESS ... | DESKTOP-9HF98IA\Sunil ... | amazon | 00:00:00 | 5 rows

```
--42.)write a query to print emp name, salary and dep id of highest salaried employee
overall.

select * from employee
where salary = (select max(salary) from employee);
```



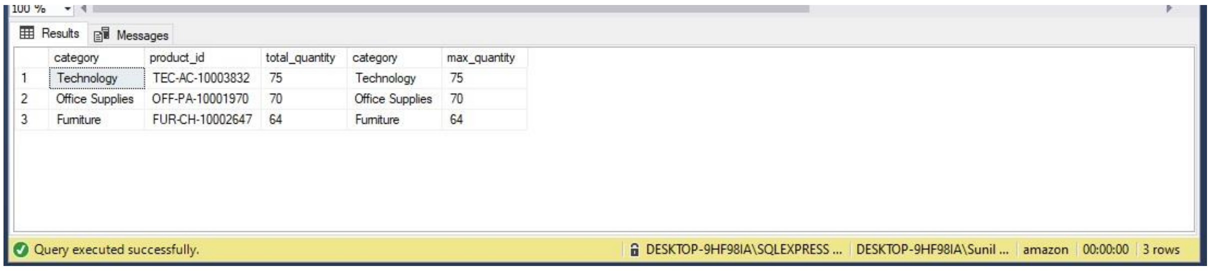
100 %

Results | Messages

	emp_id	emp_name	dept_id	salary	manager_id	emp_age
1	2	Mohit	100	15000	5	48

Query executed successfully. | DESKTOP-9HF98IA\SQLEXPRESS ... | DESKTOP-9HF98IA\Sunil ... | amazon | 00:00:00 | 1 rows

```sql
--44.)write a query to print product id and total sales of highest selling products
--(by no of units sold) in each category

with product_quantity as (
select category,product_id,sum(quantity) as total_quantity
from orders
group by category,product_id)
,cat_max_quantity as (
select category,max(total_quantity) as max_quantity from product_quantity
group by category
)
select *
from product_quantity pq
inner join cat_max_quantity cmq on pq.category=cmq.category
where pq.total_quantity  = cmq.max_quantity;
```



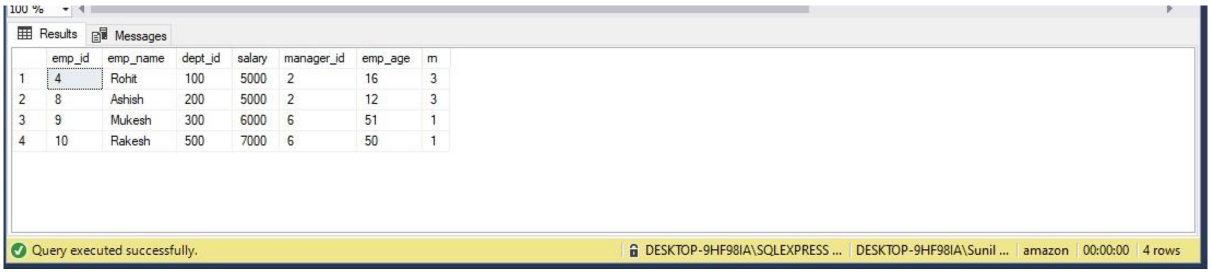
100 %

Results | Messages

	category	product_id	total_quantity	category	max_quantity
1	Technology	TEC-AC-10003832	75	Technology	75
2	Office Supplies	OFF-PA-10001970	70	Office Supplies	70
3	Furniture	FUR-CH-10002647	64	Furniture	64

Query executed successfully. | DESKTOP-9HF98IA\SQLEXPRESS ... | DESKTOP-9HF98IA\Sunil ... | amazon | 00:00:00 | 3 rows

```sql
--45.)write a query to print 3rd highest salaried employee details for each department
--(give preferece to younger employee in case of a tie).
--In case a department has less than 3 employees then print the details of highest
salaried employee
--in that department.


with rnk as (
select *, dense_rank() over(partition by dept_id order by salary desc) as rn
from employee)
,cnt as (select dept_id,count(1) as no_of_emp from employee group by dept_id)
select
rnk.*
from
rnk
inner join cnt on rnk.dept_id=cnt.dept_id
where rn=3 or  (no_of_emp<3 and rn=1);
```



100 %

Results | Messages

	emp_id	emp_name	dept_id	salary	manager_id	emp_age	rn
1	4	Rohit	100	5000	2	16	3
2	8	Ashish	200	5000	2	12	3
3	9	Mukesh	300	6000	6	51	1
4	10	Rakesh	500	7000	6	50	1

Query executed successfully. | DESKTOP-9HF98IA\SQLEXPRESS ... | DESKTOP-9HF98IA\Sunil ... | amazon | 00:00:00 | 4 rows