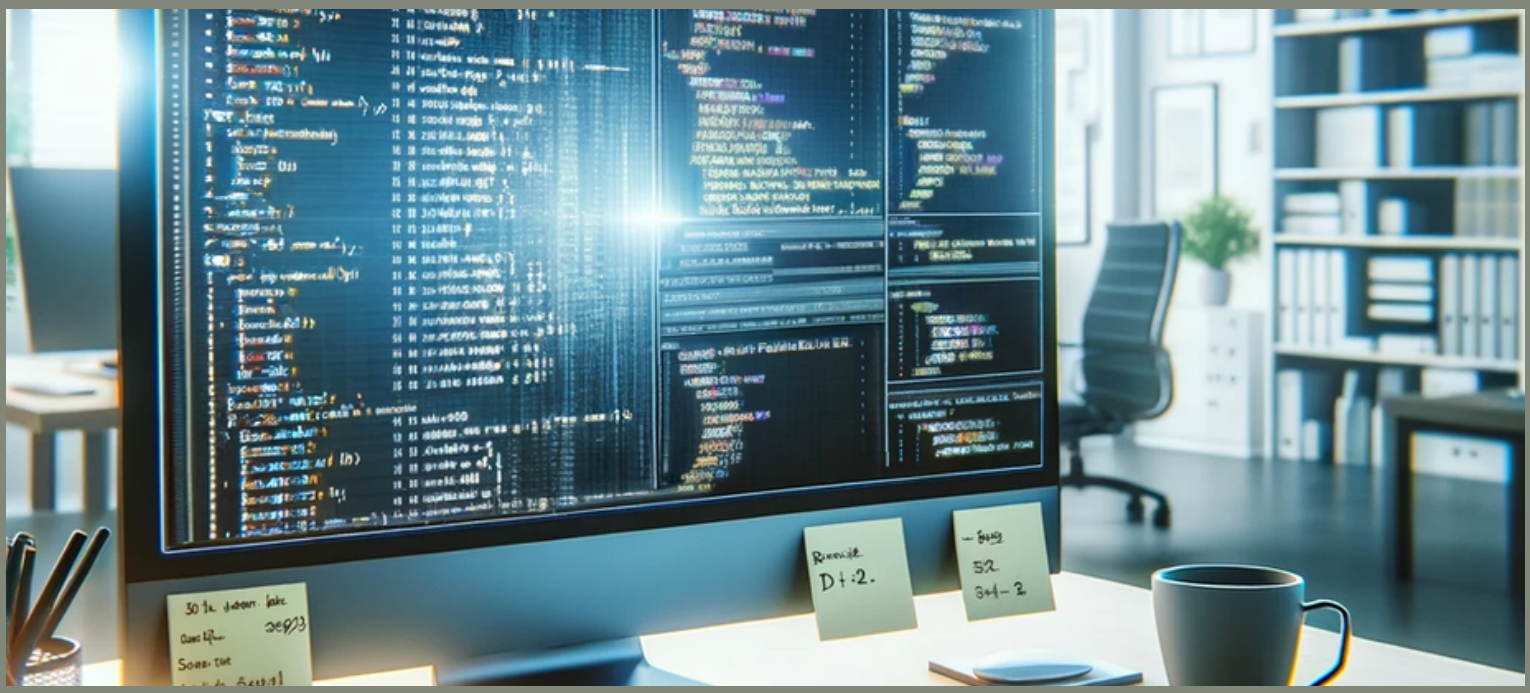
atomcamp ATOMCAMP

31 | 01|2024

SQL Data Analysis Project



muhammadhamidkhan2224@gmail.com

Table of Contents

Objectives

Briefly explained

Analysis of Customer Service Usage Patterns:

To examine and characterize customer service usage patterns, including data and minutes used across different service types (e.g., broadband, streaming, mobile), with an aim to identify trends and anomalies in service consumption.

Evaluation of Customer Feedback and Satisfaction:

To assess customer feedback in relation to different services, categorizing feedback based on service type and rating levels. This objective seeks to understand customer satisfaction and its relationship with specific services or service attributes.

Billing and Subscription Trends Analysis:

To analyse customer billing and subscription data, focusing on payment behaviours (like timeliness and status of payments) and subscription categorization (new vs. old subscribers), to gain insights into customer financial interactions and subscription preferences.

Aggregating Data

Exercise: Find the average monthly rate for each service type in service_packages. Use the ROUND function here to make result set neater.

Code:

```
select service_type , Round(AVG(monthly_rate),0) as average
from Service_Packages
GROUP BY service_type;
```

Service_Type	Average
Broadband	55
Streaming	54
Mobile	56

Exercise: Identify the customer who has used the most data in a single service_usage record. (covers ORDER BY and LIMIT that we did in last class)

Code:

```
SELECT TOP 1 *FROM service_usage
ORDER BY data_used DESC;
```

Usage_ID	Customer_ID	Service_Type	Data_Used	Minutes_Used	Usage_Date
572	231	Broadband. Streaming	998.619995117188	7578	2023-11-25

Exercise: Calculate the total minutes used by all customers for mobile services.

Code:

```
SELECT customer_id, SUM(minutes_used) as total_time_used
FROM Service_Usage
WHERE service_type = 'mobile'
GROUP BY customer_id
order by total_time_used asc;
```

70 %

Results | Messages

	customer_id	total_time_used
1	69	37
2	941	72
3	759	79
4	426	260
5	320	270
6	437	287
7	201	317
8	682	346
9	877	367
10	27	372
11	212	378
12	68	381
13	65	416
14	368	439
15	889	453
16	674	456
17	609	543
18	970	611
19	300	617
20	395	679
21	347	748
22	348	856
23	122	902
24	197	954
25	61	1034
26	366	1082
27	604	1160
28	216	1164
29	184	1197
30	914	1293
31	157	1295
32	642	1308
33	181	1361
34	119	1387
35	329	1389
36	742	1461
37	45	1491
38	70	1520

Exercise: List the total number of feedback entries for each rating level.

Code:

```
select customer_id, count(rating) as Total_num_feedback
from Feedback
GROUP BY customer_id;
```

Results | Messages

	customer_id	Total_num_feedback
1	1	2
2	2	1
3	3	3
4	5	1
5	7	2
6	8	2
7	9	1
8	10	1
9	11	1
10	13	2
11	19	1
12	20	3
13	21	1
14	22	2
15	26	1
16	27	2
17	30	1
18	32	3
19	33	0
20	34	2
21	35	1
22	37	1
23	38	1
24	40	1
25	45	2
26	46	1
28	50	2
29	51	1
30	52	2
31	54	3
32	55	2
33	56	2
34	58	1
35	59	2
36	60	4
37	61	1
38	63	1
39	64	1
40	65	1

Exercise: Calculate the total data and minutes used per customer, per service type.

Code:

```
select  customer_id, round(sum(data_used),0) as total_data_used,
sum(minutes_used) as total_minutes_used
from Service_Usage
group by customer_id
order by total_minutes_used asc;
```

Results | Messages

	customer_id	total_data_used	total_minutes_used
1	69	687	37
2	941	41	72
3	759	432	79
4	180	948	114
5	704	853	117
6	520	638	177
7	361	525	216
8	984	399	229
9	426	164	260
10	320	205	270
11	437	802	287
12	450	444	296
13	682	890	346
14	486	213	358
15	11	432	362
16	877	208	367
17	27	842	372
18	212	437	378
19	767	265	393
20	65	153	416
21	889	561	453
22	674	178	456
23	147	880	463
24	932	625	483
25	653	970	486
26	609	45	543

Query executed successfully.

Exercise: Group feedback by service impacted and rating to count the number of feedback entries.

Code:

```
select Top 10  customer_id ,round(sum(rating),0) as total_rating
from Feedback
group by  customer_id;
```

100 %

Results | Messages

	customer_id	total_rating
1	1	6
2	2	5
3	3	14
4	5	2
5	7	6
6	8	5
7	9	3
8	10	1
9	11	4
10	13	3

HAVING clause

Exercise: Show the total amount due by each customer, but only for those who have a total amount greater than $100.

Code:

```
select Customer_id, SUM(cast(amount_due as decimal(10,2))) as total_due_amount
from Billing
group by Customer_id
having SUM(cast(amount_due as decimal(10,2))) > 100;
```

100 %

Results | Messages

	Customer_id	total_due_amount
1	1	6203.74
2	2	3176.55
3	3	1516.05
4	4	8167.78
5	5	4154.27
6	6	4655.14
7	7	5858.68
8	8	2321.34
9	9	1618.28
10	10	382.97
11	11	650.19
12	12	8519.83
13	13	3950.59
14	14	1523.82

Query executed successfully.

	Customer_id	total_due_amount
14	14	1523.82
15	15	7637.33
16	16	4999.39
17	17	1061.55
18	18	7825.14
19	19	2631.32
20	20	2308.61
21	21	4354.43
22	22	1162.86
23	23	3511.94
24	24	8785.03
25	25	966.36
26	26	5635.99
27	27	9663.54

Query executed successfully.

Determine which customers have provided feedback on more than one type of service, but have a total rating of less than 10.

Code:

```
select rating,count(*) service_impacted
from Feedback
group by rating
having rating < 10;
```

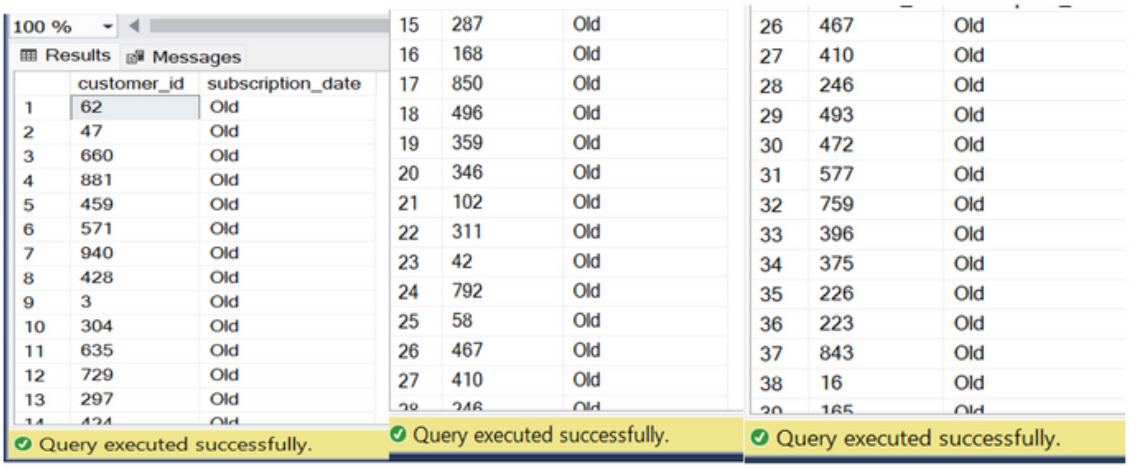
Rating	Service_impacted
3	203
1	211
4	194
5	190
2	167

Conditional Expressions and CASE Statements

Exercise: Categorize customers based on their subscription date: ‘New’ for those subscribed after 2023-01-01, ‘Old’ for all others.

Code:

```
select customer_id,
case when start_date > '2023-01-01' then 'New'
else 'Old'
end as subscription_date
from Subscriptions;
```



100 %

Results | Messages

	customer_id	subscription_date
1	62	Old
2	47	Old
3	660	Old
4	881	Old
5	459	Old
6	571	Old
7	940	Old
8	428	Old
9	3	Old
10	304	Old
11	635	Old
12	729	Old
13	297	Old
14	424	Old

Query executed successfully.

15	287	Old
16	168	Old
17	850	Old
18	496	Old
19	359	Old
20	346	Old
21	102	Old
22	311	Old
23	42	Old
24	792	Old
25	58	Old
26	467	Old
27	410	Old
28	246	Old

Query executed successfully.

26	467	Old
27	410	Old
28	246	Old
29	493	Old
30	472	Old
31	577	Old
32	759	Old
33	396	Old
34	375	Old
35	226	Old
36	223	Old
37	843	Old
38	16	Old
39	165	Old

Query executed successfully.

Exercise: Provide a summary of each customer's billing status, showing 'Paid' if the payment_date is not null, and 'Unpaid' otherwise.

Code:

```
select customer_id,
case when payment_date is not null then 'Paid'
else 'Unpaid'
end as Billing_status
from Billing;
```

100 %

Results | Messages

	customer_id	Billing_status
1	100	Paid
2	101	Paid
3	102	Paid
4	103	Paid
5	104	Paid
6	105	Paid
7	106	Unpaid
8	107	Paid
9	108	Paid
10	109	Unpaid
11	110	Paid
12	111	Paid
13	112	Paid
14	113	Paid
15	114	Paid
16	115	Paid
17	116	Paid
18	117	Unpaid
19	118	Paid
20	119	Paid
21	120	Paid
22	121	Paid
23	122	Paid
24	123	Unpaid
25	124	Paid
26	125	Paid
27	126	Paid
28	127	Paid
29	128	Paid
30	129	Paid
31	130	Paid
32	131	Paid
33	132	Paid
34	133	Paid
35	134	Paid
36	135	Paid
37	136	Paid
38	137	Unpaid

Query executed successfully.

Categorize into low , moderate and high

Code:

```
select customer_id,data_used,
case when data_used <= 100 then 'Low usage'
when data_used <= 1000 then 'Moderate usage'
when data_used <= 10000 then 'High usage'
else 'No category '
end as Data_labels
from Service_Usage;
```

100 %

Results | Messages

	customer_id	data_used	Data_labels
1	178	245.029998779297	Moderate usage
2	594	140.339996337891	Moderate usage
3	61	703.559997558594	Moderate usage
4	670	105.889999389648	Moderate usage
5	632	110.160003662109	Moderate usage
6	100	361.260009765625	Moderate usage
7	706	819.950012207031	Moderate usage
8	466	458.809997558594	Moderate usage
9	853	767.049987792969	Moderate usage
10	217	45.2299995422363	Low usage
11	543	187.779998779297	Moderate usage
12	374	354.970001220703	Moderate usage
13	562	937.52001953125	Moderate usage
14	433	636.719970703125	Moderate usage
15	171	136.309997558594	Moderate usage
16	671	996.659973144531	Moderate usage
17	153	224.800003051758	Moderate usage
18	84	156.259994506836	Moderate usage
19	478	829.419982910156	Moderate usage
20	185	922.739990234375	Moderate usage
21	819	944.890014648438	Moderate usage
22	669	637.5	Moderate usage
23	46	283.420013427734	Moderate usage
24	333	423.690002441406	Moderate usage
25	514	38.4199981689453	Low usage
26	425	507.880004882813	Moderate usage

Query executed successfully.

Exercise: In service_usage, label data usage as ‘High’ if above the average usage, ‘Low’ if below.

Code:

```
select round(avg(data_used),0) as Average_usage from Service_Usage;

select customer_id, data_used,
case when data_used > 501 then 'High'
else 'Low'
end as Caterory_of_data_usage
from service_usage;
```

100 %

Results | Messages

	Average_usage
1	501

	customer_id	data_used	Caterory_of_data_usage
1	178	245.029998779297	Low
2	594	140.339996337891	Low
3	61	703.559997558594	High
4	670	105.889999389648	Low
5	632	110.160003662109	Low
6	100	361.260009765625	Low
7	706	819.950012207031	High
8	466	458.809997558594	Low
9	853	767.049987792969	High
10	217	45.2299995422363	Low

Query executed successfully.

10	217	45.2299995422363	Low
11	543	187.779998779297	Low
12	374	354.970001220703	Low
13	562	937.52001953125	High
14	433	636.719970703125	High
15	171	136.309997558594	Low
16	671	996.659973144531	High
17	153	224.800003051758	Low
18	84	156.259994506836	Low
19	478	829.419982910156	High
20	185	922.739990234375	High
21	819	944.890014648438	High
22	669	637.5	High

Query executed successfully.

Exercise: For each feedback given, categorize the service_impacted into ‘Digital’ for ‘streaming’ or ‘broadband’ and ‘Voice’ for ‘mobile’.

Code:

```
select customer_id, service_impacted as Impacted,
case
when service_impacted = 'Streaming' then 'Digital'
when service_impacted = 'Mobile' then 'Voice'
else service_impacted
end as Service_categories
from Feedback;
```

100 %

Results | Messages

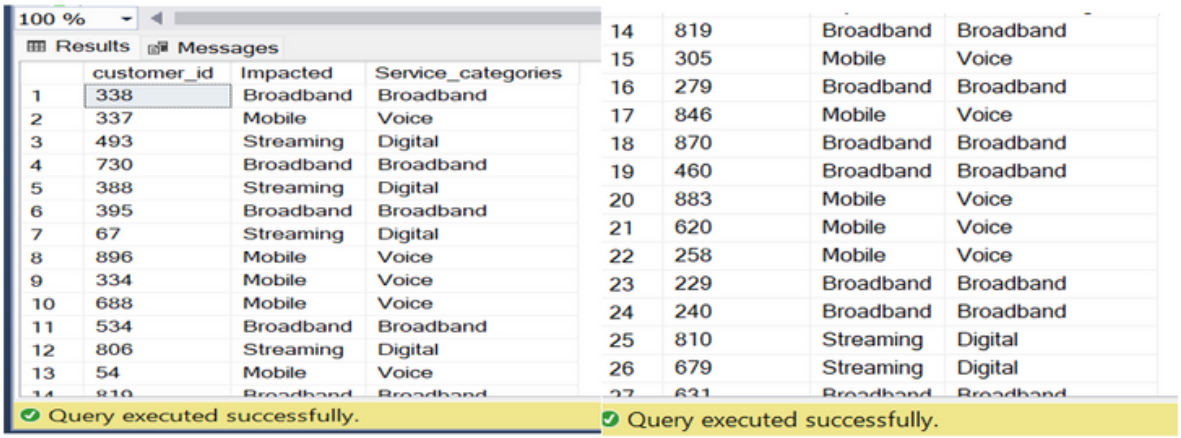
	customer_id	Impacted	Service_categories
1	338	Broadband	Broadband
2	337	Mobile	Voice
3	493	Streaming	Digital
4	730	Broadband	Broadband
5	388	Streaming	Digital
6	395	Broadband	Broadband
7	67	Streaming	Digital
8	896	Mobile	Voice
9	334	Mobile	Voice
10	688	Mobile	Voice
11	534	Broadband	Broadband
12	806	Streaming	Digital
13	54	Mobile	Voice
14	819	Broadband	Broadband

Query executed successfully.

Exercise: Update the discounts_applied field in billing to 10% of amount_due for bills with a payment_date past the due_date, otherwise set it to 5%.

Code:

```
UPDATE Billing
SET payment_date =
   CASE
      WHEN payment_date <> '' THEN CONVERT(datetime, payment_date, 101)
      ELSE null
   END;
 select *from Billing;
```



Exercise: Classify each customer as ‘High Value’ if they have a total amount due greater than $500, or ‘Standard Value’ if not.

Code:

```
select round((amount_due),0),customer_id,
case
when amount_due >= 500 then 'High Value'
when amount_due <= 500 then 'Stantard Value'
else 'other'
end as status_of_amount_due
from Billing;
```

100 %

Results | Messages

	(No column name)	customer_id	status_of_amount_due
1	8262	100	High Value
2	1695	101	High Value
3	9368	102	High Value
4	1101	103	High Value
5	1041	104	High Value
6	6754	105	High Value
7	2738	106	High Value
8	5566	107	High Value
9	3644	108	High Value
10	9717	109	High Value
11	9839	110	High Value
12	2108	111	High Value
13	7501	112	High Value
14	9465	113	High Value

Query executed successfully.

14	9465	113	High Value
15	2846	114	High Value
16	7469	115	High Value
17	9116	116	High Value
18	6992	117	High Value
19	9010	118	High Value
20	6995	119	High Value
21	4718	120	High Value
22	2955	121	High Value
23	7611	122	High Value
24	376	123	Stantard Value
25	2336	124	High Value
26	4473	125	High Value
27	5457	126	High Value

Query executed successfully.

Exercise: Mark each feedback entry as ‘Urgent’ if the rating is 1 and the feedback text includes ‘outage’ or ‘down’.

Code:

```
select *from Feedback;
 ALTER TABLE Feedback
ADD feedback_status VARCHAR(10);
UPDATE Feedback
set feedback_status=
case when rating = 1 and (lower(feedback_text) like '%outrage%' or lower(feedback_text) like '%down%') then 'urgent'
else 'normal'
end;
```

100 %

Results | Messages

	feedback_id	customer_id	feedback_text	feedback_date	service_impacted	rating	feedback_status
1	1	338	Neon's 24/7 customer support has never let me do...	2023-02-13	Broadband	4	normal
2	2	337	no matter the hour.	2023-05-10	Mobile	4	normal
3	3	493	making it difficult to manage my account on the go.	2022-11-10	Streaming	1	normal
4	4	730	I've enjoyed blazing-fast internet speeds on all my d...	2023-03-20	Broadband	2	normal
5	5	388	I've enjoyed blazing-fast internet speeds on all my d...	2023-10-09	Streaming	NULL	normal
6	6	395	I'm thrilled with Neon's rewards program—it's refres...	2023-05-31	Broadband	1	normal
7	7	67	and there are no significant benefits for long-term c...	2022-12-11	Streaming	3	normal
8	8	896	no matter the hour.	2023-01-05	Mobile	1	normal
9	9	334	I've had recurring issues with Neon's mobile app—it ...	2023-04-18	Mobile	2	normal
10	10	688	Neon's response to my lost phone situation was ab...	2023-04-12	Mobile	2	normal
11	11	534	Neon's family plan offers incredible value. We've sav...	2023-05-21	Broadband	3	normal
12	12	806	no matter the hour.	2023-08-29	Streaming	3	normal
13	13	54	The recent transition to Neon's new billing system w...	2023-03-29	Mobile	5	normal
14	14	819	Ever since Neon updated their network in my area ...	2023-09-06	Broadband	2	normal

Query executed successfully. DESKTOP-DT8T0QC\SQLEXPRESS ...

Exercise: In billing, create a flag for each bill that is ‘Late’ if the payment_date is after the due_date, ‘On-Time’ if it’s the same, and ‘Early’ if before.

Code:

```
select *from Billing;
alter table billing
add payment_status VARCHAR(10);

update Billing
set payment_status=
case
when payment_date > due_date then 'Late'
when  payment_date < due_date then 'Early'
when payment_date = due_date  then 'On Time'
else 'Unkown'
end;
```

100 %

Results | Messages

	bill_id	Customer_id	amount_due	due_date	payment_date	billing_cycle	discounts_applied	late_fee	payment_status
1	1	100	8262.080078125	2023-05-04	2023-12-22	23-Nov	103.709999084473	955.900024414063	Late
2	2	101	1694.7099609375	2023-09-17	2023-08-17	23-Jun	609.809997558594	487.070007324219	Early
3	3	102	9367.5302734375	2023-11-04	2023-09-25	23-Jan	202.520004272461	172.199996948242	Early
4	4	103	1101.07995605469	2023-10-17	2023-11-01	23-Jun	715.539978027344	25.0900001525879	Late
5	5	104	1041.16003417969	2023-10-02	2023-03-13	23-Jan	272.510009765625	481.269989013672	Early
6	6	105	6753.58984375	2023-06-21	2023-01-13	23-Apr	891.880004882813	846.890014648438	Early
7	7	106	2737.5	2023-06-26	NULL	23-Jun	15.5100002288818	605.700012207031	Unkown
8	8	107	5566.259765625	2023-01-28	2023-02-21	23-Jun	299.440002441406	65.4499969482422	Late
9	9	108	3644.35009765625	2023-06-05	2023-11-01	23-Dec	307.980010986328	682.409973144531	Late
10	10	109	9717.41015625	2023-09-04	NULL	23-Feb	359.25	843.359985351563	Unkown
11	11	110	9839.240234375	2023-01-25	2023-01-24	23-Apr	588.77001953125	158.899993896484	Early
12	12	111	2107.80004882813	2023-03-16	2023-05-20	23-Dec	869.619995117188	135.350006103516	Late
13	13	112	7500.85009765625	2023-04-14	2023-11-29	23-Mar	693.460021972656	36.189998626709	Late
14	14	113	9464.86035156625	2023-06-04	2023-03-24	23-Jun	123.700000015527	706.400024414063	Early

Query executed successfully. DESKTOP-DT8T0QC\SQLEXPRESS ...

Conclusion

The SQL profile project encompasses a variety of exercises aimed at enhancing skills in data manipulation, aggregation, and analysis using SQL. Exercises include calculating average service rates, identifying high data usage, and categorizing customer feedback, showcasing the practical application of SQL commands in real-world scenarios.

Through these exercises, learners gain hands-on experience with key SQL functionalities such as ROUND, GROUP BY, ORDER BY, and conditional expressions like CASE statements. This approach not only reinforces SQL syntax but also deepens understanding of how to extract and interpret data insights effectively.

The project further explores advanced SQL techniques, including data categorization based on specific criteria, updating records based on conditional logic, and creating flags to highlight particular data points. This comprehensive exploration equips learners with the necessary tools to tackle complex data analysis challenges, preparing them for real-world data handling and decision-making processes.
