

DMDD FINAL PROJECT

TOPIC: Smartphone Specification and Price Classification System.

TEAM MEMBERS:

VATSAL MEHTA (NUID- 002912412)

VIDIP KAMDAR (NUID- 002701593)

HARSH JAIN (NUID- 002747565)

HRUSHITHA PUTTALA (NUID- 002795117)

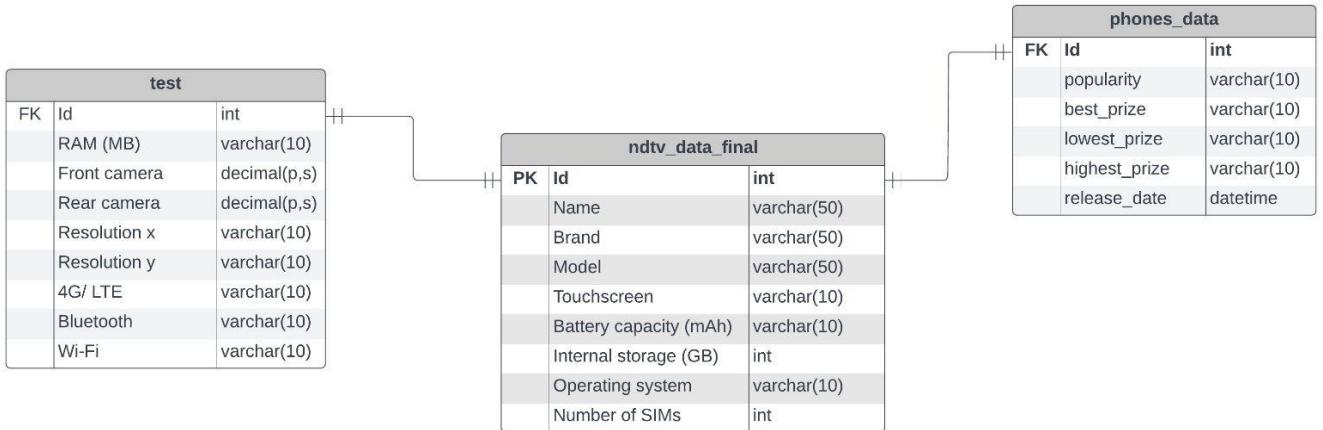
GITHUB REPOSITORY LINK:

<https://github.com/Vatsal-77/Smartphone-specification-and-price-classification-system>

Project objective:

The main purpose of the project is to create a database with respect to smart phones available in the market. The specifications mentioned in the database includes the names of the Brand, model, price, storage, resolution, dual sim availability, battery capacity and few other attributes. Since there are a vast number of smartphones available in the market, there is a possibility for the customer to be confused on what he wants to purchase. With this database which includes all the details about different mobile phones, it gets easier for the customer to choose which smartphone to buy according to his needs/requirements. Moreover, the database will also be beneficial for the corporate companies associated with the manufacturing of smartphones. The sales and marketing department of various brands can keep a track of the products in demand, accordingly the products customers show no interest in. On a global level various companies can be up to date with current market trends and hold their ground in such an altering market of the present generation. The extension of the project is detecting the price change of a specific smartphone and maintaining it in the database accordingly.

ER Diagram:



Snippets From Database:

1.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
	Id	Name	Brand	Model	Battery ca	Screen size	Touchscr	Resolution	Resolution	Processor	RAM (MB)	Internal st	Rear came	Front cam	Operating	Wi-Fi	Bluetooth	GPS	Number of 3G	4G/ LTE	Price		
0	OnePlus 7T OnePlus	7T Pro Mc	4085	6.67 Yes	1440	3120	8	12000	256	48	16	Android	Yes	Yes	Yes	Yes	Yes	2 Yes	Yes	58998			
1	Realme X2 Realme	X2 Pro	4000	6.5 Yes	1080	2400	8	6000	64	64	16	Android	Yes	Yes	Yes	Yes	Yes	2 Yes	Yes	27999			
2	iPhone 11 Apple	iPhone 11	3969	6.5 Yes	1242	2688	6	4000	64	12	12	iOS	Yes	Yes	Yes	Yes	Yes	2 Yes	Yes	106900			
3	iPhone 11 Apple	iPhone 11	3110	6.1 Yes	828	1792	6	4000	64	12	12	iOS	Yes	Yes	Yes	Yes	Yes	2 Yes	No	62900			
4	LG G8X ThinQ LG	G8X ThinQ	4000	6.4 Yes	1080	2340	8	6000	128	12	32	Android	Yes	Yes	Yes	Yes	Yes	1 No	No	49990			
5	OnePlus 7T OnePlus	7T	3800	6.55 Yes	1080	2400	8	8000	128	48	16	Android	Yes	Yes	No	Yes	Yes	2 Yes	Yes	34930			
6	OnePlus 7T OnePlus	7T Pro	4085	6.67 Yes	1440	3120	8	8000	256	48	16	Android	Yes	Yes	Yes	Yes	Yes	2 Yes	Yes	52990			
7	Samsung Galaxy Note 10 Samsung	Galaxy Note 10	4300	6.8 Yes	1440	3040	8	12000	256	12	10	Android	Yes	Yes	Yes	Yes	Yes	2 Yes	Yes	76999			
8	Asus ROG Phone Asus	ROG Phone	6000	6.59 Yes	1080	2340	8	8000	128	48	24	Android	Yes	Yes	Yes	Yes	Yes	1 Yes	Yes	37999			
9	Xiaomi Redmi K20 Xiaomi	Redmi K20	4000	6.39 Yes	1080	2340	8	6000	128	48	20	Android	Yes	Yes	Yes	Yes	Yes	2 No	No	23190			
10	Oppo K3 Oppo	K3	3765	6.5 Yes	1080	2340	8	6000	64	16	16	Android	Yes	Yes	Yes	Yes	Yes	2 Yes	Yes	23990			
11	Realme X Realme	X	3765	6.53 Yes	1080	2340	8	4000	128	48	16	Android	Yes	Yes	Yes	Yes	Yes	2 Yes	Yes	14999			
12	Xiaomi Redmi K20 Xiaomi	Redmi K20	4000	6.39 Yes	1080	2340	8	6000	64	48	20	Android	Yes	Yes	Yes	Yes	Yes	2 Yes	Yes	19282			
13	OnePlus 7 OnePlus	7 Pro	4000	6.67 Yes	1440	3120	8	6000	128	48	16	Android	Yes	Yes	Yes	Yes	Yes	2 Yes	Yes	39995			
14	Oppo Reno 10x Zoom Oppo	Reno 10x Z	4065	6.6 Yes	1080	2340	8	6000	128	48	16	Android	Yes	Yes	Yes	Yes	Yes	2 Yes	Yes	36990			
15	Realme 3 Realme	3 Pro	4045	6.3 Yes	1080	2340	8	4000	64	16	25	Android	Yes	Yes	Yes	Yes	Yes	2 Yes	Yes	13999			
16	Huawei P30 Huawei	P30 Pro	4200	6.47 Yes	1080	2340	8	8000	256	40	32	Android	Yes	Yes	No	Yes	Yes	2 Yes	Yes	54280			
17	Redmi Note 8 Xiaomi	Redmi Note 8	4000	6.3 Yes	1080	2340	8	4000	64	48	13	Android	Yes	Yes	Yes	Yes	Yes	2 Yes	Yes	9799			
18	Huawei Mate 20 Pro Huawei	Mate 20 Pro	4200	6.39 Yes	1440	3120	8	6000	128	40	24	Android	Yes	Yes	Yes	Yes	Yes	2 Yes	Yes	63990			
19	LG V40 ThinQ LG	V40 ThinQ	3300	6.4 Yes	1440	3120	8	6000	128	12	8	Android	Yes	Yes	Yes	Yes	Yes	2 Yes	Yes	29999			
20	OnePlus 6T OnePlus	6T	3700	6.41 Yes	1080	2340	8	6000	128	16	16	Android	Yes	Yes	Yes	Yes	Yes	2 Yes	Yes	31999			
21	Apple iPhone 11 Pro Apple	iPhone XR	2942	6.1 Yes	828	1792	6	3000	64	12	7	iOS	Yes	Yes	Yes	Yes	Yes	2 Yes	Yes	45499			
22	Apple iPhone 11 Pro Apple	iPhone XS	2658	6.5 Yes	1242	2688	6	4000	64	12	7	iOS	Yes	Yes	Yes	Yes	Yes	2 Yes	Yes	69999			
23	Apple iPhone 11 Pro Apple	iPhone XS	2658	5.8 Yes	1125	2436	6	4000	64	12	7	iOS	Yes	Yes	Yes	Yes	Yes	2 Yes	Yes	59999			
24	Google Pixel 3 Google	Pixel 3 XL	3430	6.3 Yes	1440	2960	8	4000	64	12,2	8	Android	Yes	Yes	Yes	Yes	Yes	1 Yes	Yes	47990			
25	Google Pixel 3 Google	Pixel 3	2915	5.5 Yes	1080	2160	8	4000	64	12,2	8	Android	Yes	Yes	Yes	Yes	Yes	1 Yes	Yes	37999			
26	Asus ROG Phone Asus	ROG Phone	4000	6 Yes	1080	2160	8	8000	128	12	8	Android	Yes	Yes	Yes	Yes	Yes	2 Yes	Yes	69999			

This is a cleaned data.

2.

	brand_name	model_name	os	popularity	best_price	lowest_price	highest_price	sellers_amount	screen_size	memory_size	battery_size	release_date
0	ALCATEL	1 1/8GB Bl	Android	422	1690	1529	1819	36	5	8	2000	Oct-20
1	ALCATEL	1 5033D 1/	Android	323	1803	1659	2489	36	5	16	2000	Sep-20
2	ALCATEL	1 5033D 1/	Android	299	1803	1659	2489	36	5	16	2000	Sep-20
3	ALCATEL	1 5033D 1/	Android	287	1803	1659	2489	36	5	16	2000	Sep-20
4	Nokia	1.3 1/16G	Android	1047	1999			10	5.71	16	3000	Apr-20
5	Honor	10 6/64GB	Android	71	10865	10631	11099	2	5.8	64	3400	Jun-18
6	Honor	10 Lite 3/3	Android	424	3999			2	6.21	32	3400	Dec-18
7	Honor	10 Lite 4/6	Android	134	4973	4733	5295	6	6.21	64	3400	Jan-19
8	Honor	10 lite 3/1:	Android	477	5100	4990	5222	3	6.21	128	3400	Jan-21
9	Honor	10 lite 3/6-	Android	215	4948	4646	5372	8	6.21	64	3400	Dec-18
10	Honor	10 lite 3/6-	Android	179	4948	4646	5372	8	6.21	64	3400	Dec-18
11	Honor	10 lite 3/6-	Android	437	5165	4897	5559	7	6.21	64	3400	Dec-18
12	Nokia	105 DS 2019 Pink	(16)	18	583	528	649	28	1.77	0.004	800	Oct-19
13	Nokia	105 Dual Sim 2019 Bla	ck	262	601	539	703	31	1.77		800	Nov-19
14	Nokia	105 Dual Sim 2019 Bla	ck	274	601	539	703	31	1.77		800	Nov-19
15	Nokia	105 Dual Sim New Bla	ck	507	552	499	606	2	1.8	0.004	800	Sep-17
16	Nokia	105 Single Sim 2019 B	lack	805	536	449	642	31	1.77		800	Nov-19
17	Nokia	105 Single Sim 2019 P	ink	692	547	496	709	35	1.77		800	Nov-19
18	Nokia	105 Single Sim New B	lack	460	663	499	1169	5	1.8	0.004	800	Sep-17
19	Nokia	105 Single Sim New B	lack	455	663	499	1169	5	1.8	0.004	800	Sep-17
20	Nokia	105 Single Sim New B	lack	378	474	450	499	2	1.8	0.004	800	Sep-17
21	Nokia	105 Single Sim New W	hite	59	499			1	1.8	0.004	800	Sep-17
22	Nokia	106 New DS Grey (16)	GB	838	596	544	699	35	1.8		800	Jan-19
23	Nokia	106 New DS Grey (16)	GB	841	596	544	699	35	1.8		800	Jan-19
24	Honor	10i 4/128G	Android	532	5499			2	6.21	128	3400	May-19
25	Honor	10i 4/128G	Android	444	5499			2	6.21	128	3400	May-19
26	Nokia	110 Dual Sim 2019 Bla	ck	471	710	640	1164	34	1.77		800	Nov-19

This is uncleaned data which we cleaned further.

Cleaning:

```
In [6]: import pandas as pd
import numpy as np
import seaborn as sns
```

```
In [7]: df = pd.read_csv("phones_data.csv")
```

```
In [8]: df.isnull().sum()
```

```
Out[8]: Unnamed: 0      0
brand_name      0
model_name      0
os             197
popularity      0
best_price      0
lowest_price    260
highest_price   260
sellers_amount   0
screen_size      2
memory_size     112
battery_size     10
release_date      0
dtype: int64
```

```
In [9]: df1 = df.dropna()
```

```
In [11]: df1.isnull().sum()
```

```
Out[11]: Unnamed: 0      0  
brand_name        0  
model_name        0  
os                0  
popularity        0  
best_price         0  
lowest_price       0  
highest_price      0  
sellers_amount     0  
screen_size        0  
memory_size        0  
battery_size       0  
release_date       0  
dtype: int64
```

```
In [12]: df1.to_csv(r'C:\Users\Harsh\Desktop\DMDD Ass3\output.csv', index=False, header=True)
```

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Id	brand_name	model_name	os	popularity	best_price	lowest_price	highest_price	sellers_amount	screen_size	memory_size	battery_size	release_date		
0	ALCATEL	1 1/8GB	Bl Android	422	1690	1529	1819	36	5	8	2000	20-Oct		
1	ALCATEL	1 5033D	1/ Android	323	1803	1659	2489	36	5	16	2000	20-Sep		
2	ALCATEL	1 5033D	1/ Android	299	1803	1659	2489	36	5	16	2000	20-Sep		
3	ALCATEL	1 5033D	1/ Android	287	1803	1659	2489	36	5	16	2000	20-Sep		
5	Honor	10 6/64GB	Android	71	10865	10631	11099	2	5.8	64	3400	18-Jun		
7	Honor	10 Lite 4/6	Android	134	4973	4733	5295	6	6.21	64	3400	19-Jan		
8	Honor	10 lite 3/1	Android	477	5100	4990	5222	3	6.21	128	3400	21-Jan		
9	Honor	10 lite 3/6	Android	215	4948	4646	5372	8	6.21	64	3400	18-Dec		
10	Honor	10 lite 3/6	Android	179	4948	4646	5372	8	6.21	64	3400	18-Dec		
11	Honor	10 lite 3/6	Android	437	5165	4897	5559	7	6.21	64	3400	18-Dec		
2	Meizu	15 4/64GB	Android	166	5272	4970	5448	3	5.45	64	3000	18-May		
3	Meizu	15 4/64GB	Android	125	5272	4970	5448	3	5.45	64	3000	18-May		
4	Meizu	15 4/64GB	Android	116	5272	4970	5448	3	5.45	64	3000	18-May		
5	Meizu	16 6/128G	Android	1111	5926	5290	6526	11	6	128	3100	18-Nov		
6	Meizu	16 6/64GB	Android	1144	5231	4599	5759	19	6	64	3100	18-Nov		
7	Meizu	16X 6/128	Android	663	5649	5599	5699	2	6	128	3100	20-Mar		
8	Meizu	16Xs 6/128	Android	900	7229	6840	7786	11	6.2	128	4000	20-Nov		
9	Meizu	16th 6/64G	Android	1063	7776	7049	8196	9	6	64	3010	18-Sep		
0	ALCATEL	1B 5002H	Android	616	2408	2168	3185	31	5.5	16	3000	20-Sep		
1	ALCATEL	1SE 3/32G	Android	629	3249	2889	3299	38	6.22	32	4000	20-Oct		
2	ALCATEL	1SE 4/128G	Android	790	3877	3581	3899	35	6.22	128	4000	20-Oct		
3	ALCATEL	1SE 4/128G	Android	793	3877	3581	3899	35	6.22	128	4000	20-Oct		
4	Nokia	2.4 2/32Gf	Android	1031	3162	2999	3299	17	6.5	32	4500	20-Sep		
5	Nokia	2720 Flip E	iOS	883	2530	2395	2689	26	2.8	4	1500	19-Sep		
6	Nokia	3.4 3/64Gf	Android	1132	3899	3550	4049	24	6.39	64	4000	20-Sep		
7	OnePlus	3T 128GB	OxygenOS	235	6991	6195	9399	9	5.5	128	3400	17-Jan		
8	OnePlus	3T 128GB	OxygenOS	188	6991	6195	9399	9	5.5	128	3400	17-Jan		

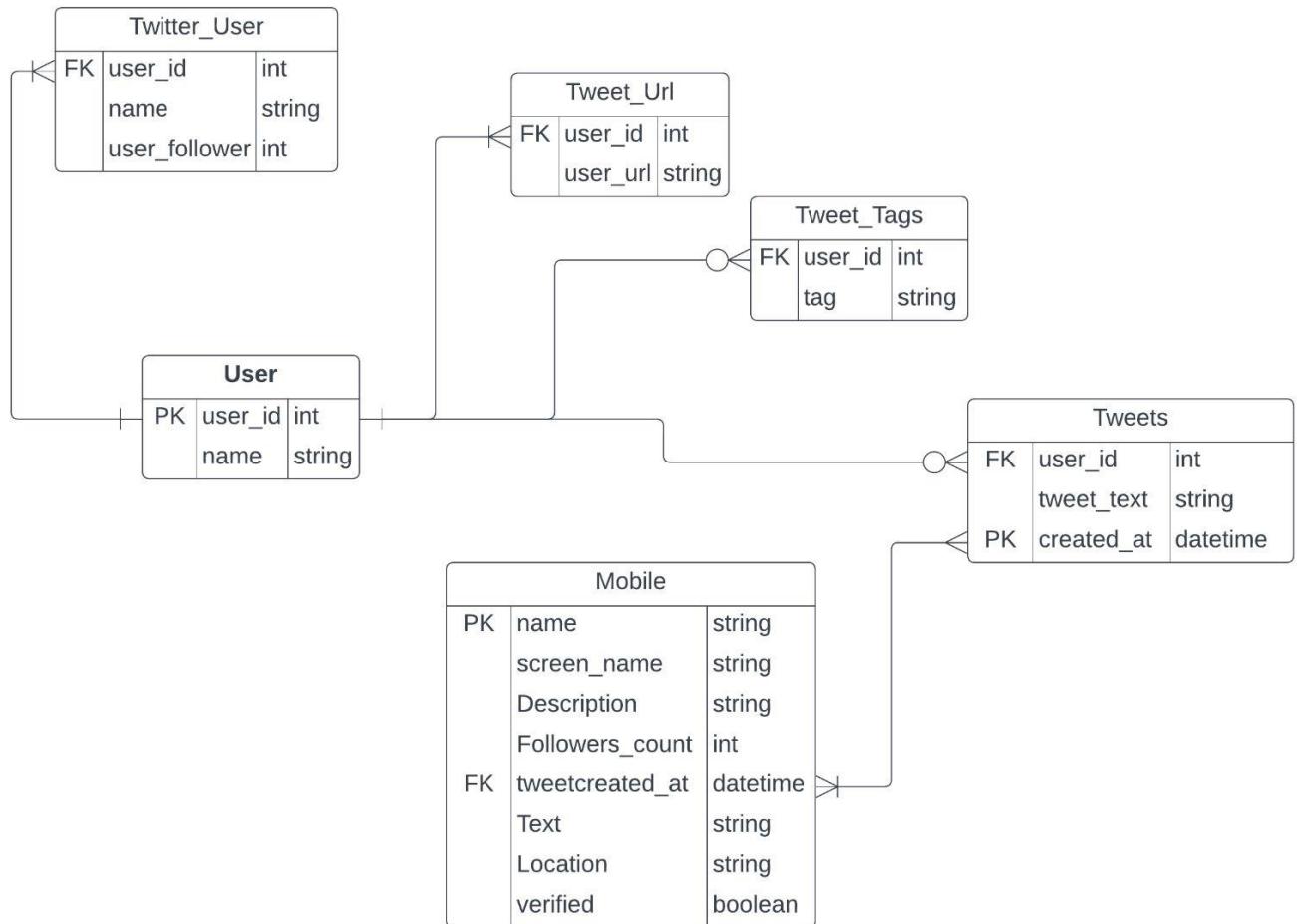
This is the data after cleaning.

3.

id	battery_px_blue	clock_spd	dual_sim	fc	four_g	int_memo	m_dep	mobile_wt	n_cores	pc	px_height	px_width	ram	sc_h	sc_w	talk_time	three_g	touch_scren	wifi	
	1	1043	1	1.8	1	14	0	5	0.1	193	3	16	226	1412	3476	12	7	2	0	1
2	841	1	0.5	1	4	1	61	0.8	191	5	12	746	857	3895	6	0	7	1	0	0
3	1807	1	2.8	0	1	0	27	0.9	186	3	4	1270	1366	2396	17	10	10	0	1	1
4	1546	0	0.5	1	18	1	25	0.5	96	8	20	295	1752	3893	10	0	7	1	1	0
5	1434	0	1.4	0	11	1	49	0.5	108	6	18	749	810	1773	15	8	7	1	0	1
6	1464	1	2.9	1	5	1	50	0.8	198	8	9	569	939	3506	10	7	3	1	1	1
7	1718	0	2.4	0	1	0	47	1	156	2	3	1283	1374	3873	14	2	10	0	0	0
8	833	0	2.4	1	0	0	62	0.8	111	1	2	1312	1880	1495	7	2	18	0	1	1
9	1111	1	2.9	1	9	1	25	0.6	101	5	19	556	876	3485	11	9	10	1	1	0
10	1520	0	0.5	0	1	0	25	0.5	171	3	20	52	1009	651	6	0	5	1	0	1
11	1500	0	2.2	0	2	0	55	0.6	80	7	6	503	1336	3866	13	7	20	0	1	0
12	1343	0	2.9	0	2	1	34	0.8	171	3	6	235	1671	3911	15	8	8	1	1	1
13	900	1	1.4	1	0	0	30	1	87	2	3	829	1893	439	6	2	20	1	0	0
14	1190	1	2.2	1	5	0	19	0.9	158	5	15	227	1856	992	13	0	16	1	1	0
15	630	0	1.8	0	8	1	51	0.9	193	8	9	1315	1323	2751	17	6	3	1	1	0
16	1846	1	1	0	5	1	53	0.7	106	8	7	185	1832	563	9	5	10	1	0	1
17	1985	0	0.5	1	14	1	26	1	163	2	17	613	1511	2083	13	3	14	1	1	0
18	1042	0	2.9	0	5	1	48	0.2	186	4	15	335	532	2187	9	2	5	1	0	0
19	1231	1	1.7	1	2	1	37	0.2	194	2	3	82	1771	3902	19	12	15	1	0	1
20	1488	0	2.6	0	9	0	37	0.7	189	4	20	47	559	2524	5	0	6	0	0	0
21	968	0	0.6	0	8	1	7	0.7	151	1	17	504	1930	1357	15	1	16	1	1	0
22	529	0	2.6	1	1	0	60	0.5	101	5	5	521	1591	3456	13	11	9	0	1	0
23	1558	0	1.7	1	7	0	50	0.1	115	2	10	777	1587	1641	17	0	9	0	1	1
24	533	1	0.7	1	16	0	58	0.8	97	5	18	512	1111	2322	17	3	2	0	1	0
25	1037	0	1.7	1	1	0	5	0.7	125	3	6	1194	1321	3862	17	4	7	1	1	0
26	1025	0	1.6	1	6	1	43	0.7	122	3	16	1003	1306	557	15	10	14	1	1	1
27	1858	0	3	1	0	0	17	0.6	124	4	1	575	1200	2427	16	11	13	1	1	0

Scrapped data from twitter:

ER diagram of scrapped data from twitter:



USE CASES:

Data Gathering, Munging and Cleaning:

1. Which brand have what quantity of phones?

use dmdd;

```
SELECT Brand,COUNT(Name) AS
```

No_of_PhonesFROM

ndtv_data_final

GROUP BY Brand;

The screenshot shows a SQL database interface with a query editor and a results grid. The query editor at the top contains the following SQL code:

```
1 • use dmdd;
2 • SELECT Brand,COUNT(Name) AS No_of_Phones
3   FROM ndtv_data_final
4   GROUP BY Brand;
```

The results grid below displays the count of phones for each brand:

Brand	No_of_Phones
Realme	18
Apple	17
LG	42
OnePlus	11
Samsung	101
Asus	37
Xiaomi	47
Oppo	35
Huawei	12
Google	10
Nokia	35
HTC	35
Motorola	42
Honor	33
Yu	13
Poco	2
Vivo	52
Nubia	12
Black Sh...	1
Infinix	17
Lenovo	42
Sony	28
Jio	1
Coolpad	19

The interface includes a toolbar with various icons for file operations, search, and navigation. A sidebar on the right provides links to other tools: Result Grid, Form Editor, Field Types, Query Stats, and Execution Plan. A status bar at the bottom indicates "Read Only".

2. Which model of the smartphone has what clockspeed?

```

use dmdd;
SELECT
Model,clock_spee
dFROM
ndtv_data_final
RIGHT JOIN test
ON ndtv_data_final.Id = test.Id;

```

The screenshot shows a SQL query editor interface with the following details:

- Query:**

```

1 • use dmdd;
2 • SELECT Model,clock_speed
3   FROM ndtv_data_final
4   RIGHT JOIN test
5     ON ndtv_data_final.Id = test.Id;

```
- Result Grid:**

Model	clock_speed
X2 Pro	1.8
iPhone 11 Pro Max	0.5
iPhone 11	2.8
G8X ThinQ	0.5
7T	1.4
7T Pro	2.9
Galaxy Note 10+	2.4
ROG Phone 2	2.4
Redmi K20 Pro	2.9
K3	0.5
X	2.2
Redmi K20	2.9
7 Pro	1.4
Reno 10x Zoom	2.2
3 Pro	1.8
P30 Pro	1
Redmi Note 7 Pro	0.5
Mate 20 Pro	2.9
V40 ThinQ	1.7
6T	2.6
iPhone XR	0.6
iPhone XS Max	2.6
iPhone XS	1.7
Pixel 3 XL	0.7
- Toolbar:** Includes standard icons for file operations, search, and export.
- Right Panel:** Shows navigation links for "Result Grid", "Form Editor", "Field Types", "Query Stats", and "Execution Plan".
- Status Bar:** Shows "Result 3" and "Read Only".

3. Which brand has what release dates of its phones? (cross join)

```

use dmdd;
SELECT ndtv_data_final.Brand, output.release_date
AS Release_DateFROM output
CROSS JOIN dmdd.ndtv_data_final;

```

The screenshot shows a SQL query editor interface with multiple tabs at the top labeled SQL File 7*, SQL File 8*, SQL File 9* (highlighted in blue), SQL File 10*, SQL File 11*, SQL File 12*, SQL File 13*, SQL File 14*, SQL File 15*, and SQL F*. Below the tabs is a toolbar with various icons. The main area contains a SQL query:

```

1 • use dmdd;
2 • SELECT ndtv_data_final.Brand, output.release_date AS Release_Date
3 FROM output
4 CROSS JOIN dmdd.ndtv_data_final;

```

Below the query is a "Result Grid" table with two columns: "Brand" and "Release_Date". The data shows multiple entries for the brand "Realme" with various release dates. The table has a header row and approximately 25 data rows. To the right of the table is a vertical sidebar with icons for different features: Result Grid (selected), Form Editor, Field Types, Query Stats, and Execution Plan. At the bottom right of the grid is a "Read Only" status indicator.

Brand	Release Date
Realme	20-Oct
Realme	19-Nov
Realme	18-Sep
Realme	18-Nov
Realme	18-Nov
Realme	20-Nov
Realme	18-Sep
Realme	18-Sep
Realme	20-Nov
Realme	18-Sep
Realme	17-Sep
Realme	17-Sep
Realme	16-Mar
Realme	16-Mar
Realme	17-Mar
Realme	20-Nov
Realme	20-Apr
Realme	20-Nov

4. Which smartphone has what weight, pixel height and pixel width?

use dmdd;

```

SELECT Name,mobile_wt AS Mobile_Weight,px_height As
Pixel_Height,px_width AS Pixel_WidthFROM ndtv_data_final
LEFT JOIN test
ON ndtv_data_final.Id = test.Id;

```

```

1 • use dmdd;
2 • SELECT Name,mobile_wt AS Mobile_Weight,px_height AS Pixel_Height,px_width AS Pixel_Width
3   FROM ndtv_data_final
4   LEFT JOIN test
5     ON ndtv_data_final.Id = test.Id;

```

Name	Mobile_Weight	Pixel_Height	Pixel_Width
Realme X2 Pro	193	226	1412
iPhone 11 Pro Max	191	746	857
iPhone 11	186	1270	1366
LG G8X ThinQ	96	295	1752
OnePlus 7T	108	749	810
OnePlus 7T Pro	198	569	939
Samsung Galaxy Note 10+	156	1283	1374
Asus ROG Phone 2	111	1312	1880
Xiaomi Redmi K20 Pro	101	556	876
Oppo K3	171	52	1009
Realme X	80	503	1336
Xiaomi Redmi K20	171	235	1671
OnePlus 7 Pro	87	829	1893
Oppo Reno 10x Zoom	158	227	1856
Realme 3 Pro	193	1315	1323
Huawei P30 Pro	106	185	1832
Redmi Note 7 Pro	163	613	1511
Huawei Mate 20 Pro	186	335	532
LG V40 ThinQ	194	82	1771
OnePlus 6T	189	47	559
Apple iPhone XR	151	504	1930
Apple iPhone XS Max	101	521	1591
Apple iPhone XS	115	777	1587
Google Pixel 2 XL	87	517	1111

Result 5 × Read Only

5. Which phone model has what release date? (inner join)

```

use dmdd;
SELECT
Model,release_date
FROM
ndtv_data_final
INNER JOIN output
ON ndtv_data_final.Id= output.Id;

```

The screenshot shows a SQL query editor interface with multiple tabs at the top labeled SQL File 7* through SQL File 15*. The active tab is "SQL File 11*". The query window contains the following SQL code:

```
1 • use dmdd;
2 • SELECT Model,release_date
3   FROM ndtv_data_final
4   INNER JOIN output
5     ON ndtv_data_final.Id= output.Id;
```

The results are displayed in a "Result Grid" table:

Model	release_date
X2 Pro	20-Sep
iPhone 11 Pro Max	20-Sep
iPhone 11	20-Sep
7T	18-Jun
Galaxy Note 10+	19-Jan
ROG Phone 2	21-Jan
Redmi K20 Pro	18-Dec
K3	18-Dec
X	18-Dec
P20 Pro	18-May
6	18-May
ZenFone 5Z (ZS6...	18-May
iPhone 8 Plus	18-Nov
iPhone X	18-Nov
Moto Z2 Force	20-Mar
U11	20-Nov
8	18-Sep
Galaxy S8	20-Sep
G6	20-Oct
3T	20-Oct
Galaxy C9 Pro	20-Oct
iPhone 7	20-Sep
Galaxy A8	19-Sep
✓	20-Sep

The sidebar on the right shows icons for Result Grid, Form Editor, Field Types, Query Stats, and Execution Plan. A "Read Only" status is indicated at the bottom right.

6. What is the total number of brands?

```
use dmdd;
SELECT Brand FROM
ndtv_data_final UNION
SELECT brand_name FROM output;
```

The screenshot shows the MySQL Workbench interface. At the top, there are tabs for various SQL files. Below the tabs is a toolbar with icons for file operations, search, and refresh. A dropdown menu says "Limit to 1000 rows". The main area contains a query editor with the following code:

```
1 • use dmdd;
2 • SELECT Brand FROM ndtv_data_final
3 UNION
4 SELECT brand_name FROM output;
```

Below the code is a "Result Grid" tab. The grid displays a list of phone brands. The first column is labeled "Brand" and contains the following data:

Brand
Realme
Apple
LG
OnePlus
Samsung
Asus
Xiaomi
Oppo
Huawei
Google
Nokia
HTC
Motorola
Honor
Yu
Poco
Vivo
Nubia
Black Sh...
Infinix
Lenovo
Sony
Jio
Coolpad

At the bottom right of the result grid, there is a "Read Only" status indicator.

7. What is the operating system for what number of phones?

```
use dmdd;
SELECT Operating_system, count(Id) AS
No_of_Phones FROM ndtv_data_final
GROUP BY
Operating_system
HAVING count(Id) > 5;
```

The screenshot shows a SQL Server Management Studio (SSMS) interface. At the top, there are tabs for various SQL files. Below the tabs is a toolbar with icons for file operations, search, and other utilities. A status bar at the bottom indicates "Result 5" and "Read Only".

The main area displays a query window with the following T-SQL code:

```
1 • use dmdd;
2 • SELECT Operating_system, count(Id) AS No_of_Phones
3   FROM ndtv_data_final
4   GROUP BY Operating_system
5   HAVING count(Id) > 5;
```

Below the code is a result grid titled "Result Grid" showing the following data:

Operating_system	No_of_Phones
Android	1298
iOS	17
Cyanogen	10
BlackBerry	10
Windows	19

To the right of the result grid is a vertical toolbar with icons for "Result Grid", "Form Editor", "Field Types", "Query Stats", and "Execution Plan".

8. What is the total number of phones from both the dataset?

```
use dmdd;
SELECT Model From
ndtv_data_finalUNION ALL
SELECT model_name FROM output;
```

The screenshot shows a SQL database interface with multiple tabs at the top labeled SQL File 7*, SQL File 8*, SQL File 9*, SQL File 10*, SQL File 11*, SQL File 12*, SQL File 13*, SQL File 14* (highlighted in blue), SQL File 15*, and SQL File 16*. Below the tabs is a toolbar with various icons. The main area contains a code editor with the following SQL query:

```
1 • use dmdd;
2 • SELECT Model From ndtv_data_final
3 UNION ALL
4 SELECT model_name FROM output;
```

Below the code editor is a results grid titled "Result Grid". The grid has a single column labeled "Model" and contains a list of smartphone models. The models listed are:

- X2 Pro
- iPhone 11 Pro Max
- iPhone 11
- G8X ThinQ
- 7T
- 7T Pro
- Galaxy Note 10+
- ROG Phone 2
- Redmi K20 Pro
- K3
- X
- Redmi K20
- 7 Pro
- Reno 10x Zoom
- 3 Pro
- P30 Pro
- Redmi Note 7 Pro
- Mate 20 Pro
- V40 ThinQ
- 6T
- iPhone XR
- iPhone XS Max
- iPhone XS
- Pixel 3 XL

At the bottom of the results grid, it says "Result 5" and there is a "Read Only" status indicator.

9. What is the average price of all the smartphones?

```
use dmdd;

SELECT AVG(ndtv_data_final.Price) AS
Average_Price FROM ndtv_data_final
INNER JOIN output ON ndtv_data_final.Id = output.Id;
```

```
SQL File 7* SQL File 8* SQL File 9* SQL File 10* SQL File 11* SQL File 12* SQL File 13* SQL File 14* SQL File 15* x SQL F |
```

```
1 • use dmdd;
2 • SELECT AVG(ndtv_data_final.Price) AS Average_Price
3 FROM ndtv_data_final
4 INNER JOIN output ON ndtv_data_final.Id = output.Id;
```

Average_Price
11316.2311

Result Grid | Form Editor | Field Types | Query Stats | Execution Plan

Result 5 x Read Only

10. Which brands have what number of phones with internal storage more than 128GB?

```
use dmdd;
```

```
SELECT Brand, count(Internal_storage) AS  
Phones_with_higher_storage FROM ndtv_data_final  
WHERE  
Internal_storage >= 128  
GROUP BY Brand;
```

The screenshot shows a SQL query editor interface with the following details:

- Query:**

```

1 use dmdd;
2 • SELECT Brand, count(Internal_storage) AS Phones_with_higher_storage
3 FROM ndtv_data_final
4 WHERE Internal_storage >= 128
5 GROUP BY Brand;
```
- Result Grid:**

Brand	Phones_with_higher_storage
LG	4
OnePlus	5
Samsung	16
Asus	4
Xiaomi	2
Realme	1
Oppo	11
Huawei	6
HTC	2
Vivo	9
Nubia	3
Motorola	2
Black Sh...	1
Honor	6
Nokia	1
Infinix	1
Micromax	1
Meizu	1
Panasonic	1
Tecno	1
- Toolbar:** Includes icons for file operations, search, and refresh.
- Status Bar:** Shows "Result 5" and "Read Only".
- Right Panel:** A sidebar with icons for Result Grid, Form Editor, Field Types, Query Stats, and Execution Plan.

11. Which phones have what processor count and

core capacity? use dmdd;

```
SELECT model,pc AS Processor_Count, n_cores AS
```

```
Core_CapacityFROM ndtv_data_final
```

```
LEFT OUTER JOIN test
```

```
ON ndtv_data_final.Id = test.Id
```

The screenshot shows a SQL query editor interface with multiple tabs at the top labeled SQL File 10* through SQL File 18*. The active tab is 'SQL File 17*'. Below the tabs is a toolbar with various icons for file operations and search. The main area contains a SQL script and its execution results.

```

1 • use dmdd;
2 • SELECT model,pc AS Processor_Count, n_cores AS Core_Capacity
3   FROM ndtv_data_final
4   LEFT OUTER JOIN test
5     ON ndtv_data_final.Id = test.Id

```

The results grid displays the following data:

model	Processor_Count	Core_Capacity
X2 Pro	16	3
iPhone 11 Pro Max	12	5
iPhone 11	4	3
G8X ThinQ	20	8
7T	18	6
7T Pro	9	8
Galaxy Note 10+	3	2
ROG Phone 2	2	1
Redmi K20 Pro	19	5
K3	20	3
X	6	7
Redmi K20	6	3
7 Pro	3	2
Reno 10x Zoom	15	5
3 Pro	9	8
P30 Pro	7	8
Redmi Note 7 Pro	17	2
Mate 20 Pro	15	4
V40 ThinQ	3	2
6T	20	4
iPhone XR	17	1
iPhone XS Max	5	5

Result 4 X Read Only

12. What is the highest and lowest price of the phones of each brand?

```

use dmdd;

SELECT output.Id, Name,Brand,
highest_price,lowest_priceFROM
ndtv_data_final
RIGHT OUTER JOIN output
ON ndtv_data_final.Id = output.Id

```

```

SQL File 10* SQL File 11* SQL File 12* SQL File 13* SQL File 14* SQL File 15* SQL File 16* SQL File 17* SQL File 18* ×
use dmdd;
SELECT output.Id, Name, Brand, highest_price, lowest_price
FROM ndtv_data_final
RIGHT OUTER JOIN output
ON ndtv_data_final.Id = output.Id

```

The screenshot shows the SSMS interface with a query window containing the provided SQL code. Below the code is a result grid titled "Result Grid" showing the following data:

	Id	Name	Brand	highest_price	lowest_price
1	Realme X2 Pro	Realme	2489	1659	
2	iPhone 11 Pro Max	Apple	2489	1659	
3	iPhone 11	Apple	2489	1659	
5	OnePlus 7T	OnePlus	11099	10631	
7	Samsung Galaxy Note 10+	Samsung	5295	4733	
8	Asus ROG Phone 2	Asus	5222	4990	
9	Xiaomi Redmi K20 Pro	Xiaomi	5372	4646	
10	Oppo K3	Oppo	5372	4646	
11	Realme X	Realme	5559	4897	
30	Huawei P20 Pro	Huawei	5448	4970	
31	OnePlus 6	OnePlus	5448	4970	
32	Asus ZenFone 5Z (ZS620KL)	Asus	5448	4970	
40	Apple iPhone 8 Plus	Apple	6526	5290	
41	Apple iPhone X	Apple	5759	4599	
42	Motorola Moto Z2 Force	Motorola	5699	5599	
43	HTC U11	HTC	7786	6840	
44	Nokia 8	Nokia	8196	7049	
46	Samsung Galaxy S8	Samsung	3185	2168	
47	LG G6	LG	3299	2889	
48	OnePlus 3T	OnePlus	3899	3581	
49	Samsung Galaxy C9 Pro	Samsung	3899	3581	
52	Apple iPhone 7	Apple	3299	2999	

Result 3 x Read Only

13. What is the name, popularity and battery power for each mobile phone where popularity is greater than 500?

```

use dmdd;
SELECT
    Name,popularity,battery_power
FROM ndtv_data_final
JOIN output
ON ndtv_data_final.Id =
    output.IdJOIN test
ON output.Id =
    test.Id WHERE
popularity>500;

```

The screenshot shows a SQL database interface with multiple tabs at the top labeled SQL File 12* through SQL File 20*. Below the tabs is a toolbar with various icons. The main area contains a query window with the following SQL code:

```

1 • use dmdd;
2 • SELECT Name,popularity,battery_power
3   FROM ndtv_data_final
4   JOIN output
5     ON ndtv_data_final.Id = output.Id
6   JOIN test
7     ON output.Id = test.Id
8   WHERE popularity > 500;

```

Below the code is a result grid titled "Result Grid". The grid has columns: Name, popularity, and battery_power. It displays a list of phones with their respective popularity and battery power values. The data includes:

Name	popularity	battery_power
Apple iPhone 8 Plus	1111	567
Apple iPhone X	1144	1952
Motorola Moto Z2 Force	663	822
HTC U11	900	685
Nokia 8	1063	1388
Samsung Galaxy S8	616	1411
LG G6	629	1094
OnePlus 3T	790	1653
Samsung Galaxy C9 Pro	793	916
Apple iPhone 7	1031	632
Samsung Galaxy A8	883	578
Poco X2	1132	1206
Samsung Galaxy Note ...	921	1366
Realme XT	1156	1391
Redmi Note 8 Pro	701	979
Realme 5 Pro	1139	1999
Asus 6Z	1118	1982
Samsung Galaxy M40	1117	1373
Redmi Note 7S	620	1151
Motorola One Vision	798	1650

At the bottom right of the interface, there is a vertical sidebar with icons for "Result Grid", "Form Editor", "Field Types", "Query Stats", and "Execution Plan".

14. Which phones have the mobile weight less than 100?use dmdd;

SELECT Id,Name

FROM

ndtv_data_final

WHERE EXISTS (SELECT mobile_wt FROM test WHERE ndtv_data_final.Id = test.Id AND mobile_wt < 100);

The screenshot shows a MySQL Workbench interface. At the top, there are tabs for various SQL files. Below the tabs is a toolbar with icons for file operations, search, and refresh. A status bar indicates "Limit to 2000 rows". The main area displays a query window with the following code:

```

1 • use dmdd;
2 • SELECT Id,Name
3   FROM ndtv_data_final
4 WHERE EXISTS (SELECT mobile_wt FROM test WHERE ndtv_data_final.Id = test.Id AND mobile_wt < 100);

```

Below the code is a result grid titled "Result Grid" with two columns: "Id" and "Name". The data in the grid is as follows:

Id	Name
4	LG G8X ThinQ
11	Realme X
13	OnePlus 7 Pro
24	Google Pixel 3 XL
28	LG G7+ ThinQ
30	Huawei P20 Pro
37	HTC U11+
40	Apple iPhone 8 Plus
46	Samsung Galaxy S8
60	Google Nexus 6P
61	Apple iPhone 6s Plus
66	Xiaomi Redmi Note
89	Asus 6Z
90	Samsung Galaxy M40
92	Motorola One Vision
96	Samsung Galaxy A70
100	Vivo V15 Pro
103	Samsung Galaxy M20
108	Realme U1
123	Nokia 6.1 Plus
127	Realme 1
136	Infinix Zero 5 Pro
139	Honor 9i
147	Google Pixel 3 XL

The right side of the interface has a sidebar with icons for "Result Grid", "Form Editor", "Field Types", "Query Stats", and "Execution Plan". A "Read Only" status is indicated at the bottom right.

15. The brand name APPLE has what number of sellers?use dmdd;
 SELECT ndtv_data_final.Id, Brand, sellers_amount AS
 No_of_sellersFROM ndtv_data_final
 JOIN output
 ON ndtv_data_final.Id = output.Id
 where ndtv_data_final.Brand='Apple' and output.sellers_amount<20;

The screenshot shows a MySQL Workbench interface. At the top, there are tabs for various SQL files. Below the tabs is a toolbar with icons for file operations, search, and refresh. A dropdown menu says "Limit to 2000 rows". The main area contains a query editor with the following SQL code:

```

1 • use dmdd;
2 • SELECT ndtv_data_final.Id, Brand, sellers_amount AS No_of_sellers
3   FROM ndtv_data_final
4   JOIN output
5     ON ndtv_data_final.Id = output.Id
6   where ndtv_data_final.Brand='Apple' and output.sellers_amount<20;

```

Below the query editor is a results grid titled "Result Grid". It has columns "Id", "Brand", and "No_of_sellers". The data is as follows:

	Id	Brand	No_of_sellers
▶	40	Apple	11
	41	Apple	19
	52	Apple	17
	192	Apple	10

On the right side of the interface, there is a sidebar with icons for different features: Result Grid (selected), Form Editor, Field Types, Query Stats, Execution Plan, and a "Read Only" status indicator.

16. Which of the following mobile phones are black in colour with their prices above 40000?use dmdd;

SELECT

model_name, Price

FROM

ndtv_data_final JOIN

output

ON ndtv_data_final.Id = output.Id

where output.model_name like '%Black%' and ndtv_data_final.Price>40000;

The screenshot shows a MySQL Workbench interface with a query editor and a results grid. The query editor contains the following SQL code:

```

1 • use dmdd;
2 • SELECT model_name, Price
3   FROM ndtv_data_final
4   JOIN output
5     ON ndtv_data_final.Id = output.Id
6   where output.model_name like '%Black%' and ndtv_data_final.Price>40000;

```

The results grid displays a list of mobile phone models and their prices, filtered by the condition where the model name contains 'Black' and the price is greater than 40000. The data is as follows:

model_name	Price
1 5033D 1/16GB Volcano Black (5033D-2LALUAF)	106900
1 5033D 1/16GB Volcano Black (5033D-2LALUAF)	62900
10 Lite 4/64GB Black	79699
15 4/64GB Black	49990
16 6/128GB Black	47000
16 6/64GB Black	69999
8x Max 4/64GB Black	59090
A60 Pro 3/16GB Black	54665
BV9600 Pro 6/128GB Black	50650
BV9600 Pro 6/128GB Black	56699
BV9800 6/128Gb Black	57915
Galaxy A20e SM-A202F 3/32GB Black SM-A202F...	45999
Galaxy M10 SM-M105F 2/16GB Black (SM-M105...	42021
Mix 2 M2 4/64GB Black	92999
Pixel 32GB (Quite Black)	74990
RAZR 2019 XT2000-2 Noir Black	57900
ROG Phone 3 ZS661KS 12/512GB Black	67900
Redmi Note 9T 4/128GB Nightfall Black	44990
iPhone 7 Plus 32GB Black (MNQM2)	88719

Below the results grid, there is a message: "Result 4" and "Read Only". On the right side of the interface, there is a sidebar with icons for Result Grid, Form Editor, Field Types, Query Stats, and Execution Plan.

17. Brand name, Operating systems and model name of one of the dataset were inserted into another dataset with similar attributes whose price is greater than 10000

```

use dmdd;
INSERT INTO
ndtv_data_final(Brand,Model,Operating_system)
SELECT brand_name, model_name,os FROM
output
WHERE highest_price>10000;
SELECT * FROM dmdd.ndtv_data_final;

```

The screenshot shows a MySQL Workbench interface. The top bar has tabs for various SQL files (SQL File 18*, SQL File 19*, SQL File 20*, SQL File 21*, SQL File 22*, SQL File 23*, SQL File 24*, SQL File 25*, SQL File 26*). The current tab is SQL File 23*. Below the tabs is a toolbar with icons for file operations, search, and refresh. A dropdown menu says "Limit to 2000 rows". The main area contains a query window with the following code:

```

1 • use dmdd;
2 • INSERT INTO ndtv_data_final(Brand,Model,Operating_system)
3     SELECT brand_name, model_name,os FROM output
4     WHERE highest_price>10000;
5 • SELECT * FROM dmdd.ndtv_data_final;

```

Below the code is a "Result Grid" table with the following data:

	ID	Name	Brand	Model	Battery_capacity	Screen_size	Touchscreen	Resolution_x	Resolution_y	Processor
	1353	Intex Aqua Y2 Ultra	Intex	Aqua Y2 Ultra	1400	4	Yes	480	800	4
	1354	Intex Aqua A2	Intex	Aqua A2	1500	4	Yes	480	800	4
	1355	Videcon Infinium Z51 No...	Videcon	Infinium Z51 Nova+	2000	5	Yes	480	854	4
	1356	Intex Aqua Y4	Intex	Aqua Y4	1700	4.5	Yes	480	854	2
	1357	iBall Andi4 B20	iBall	Andi4 B20	1250	4	Yes	480	800	1
	1358	iBall Andi Avonte 5	iBall	Andi Avonte 5	2150	5	Yes	480	854	4
	HULL	Honor 10 6/64GB Black	HULL	HULL	HULL	HULL	HULL	HULL	HULL	HULL
	HULL	OnePlus 6T 8/256GB Midn...	HULL	HULL	HULL	HULL	HULL	HULL	HULL	HULL
	HULL	OnePlus 7 8/256GB Mirror ...	HULL	HULL	HULL	HULL	HULL	HULL	HULL	HULL
	HULL	OnePlus 8 12/256GB Onyx...	HULL	HULL	HULL	HULL	HULL	HULL	HULL	HULL
	HULL	OnePlus 8 8/128GB Onyx ...	HULL	HULL	HULL	HULL	HULL	HULL	HULL	HULL
	HULL	OnePlus 8 Pro 12/256GB G...	HULL	HULL	HULL	HULL	HULL	HULL	HULL	HULL
	HULL	OnePlus 8 Pro 8/128GB On...	HULL	HULL	HULL	HULL	HULL	HULL	HULL	HULL
	HULL	Nokia 8.3 5G 8/128GB P...	HULL	HULL	HULL	HULL	HULL	HULL	HULL	HULL
	HULL	OnePlus 8T 12/256GB Cyb...	HULL	HULL	HULL	HULL	HULL	HULL	HULL	HULL
	HULL	OnePlus 8T 12/256GB Lun...	HULL	HULL	HULL	HULL	HULL	HULL	HULL	HULL
	HULL	OnePlus 8T 8/128GB Aqua...	HULL	HULL	HULL	HULL	HULL	HULL	HULL	HULL
	HULL	Ulefone Armor 10 5G 8/12...	HULL	HULL	HULL	HULL	HULL	HULL	HULL	HULL
	HULL	Ulefone Armor 9 8/128GB ...	HULL	HULL	HULL	HULL	HULL	HULL	HULL	HULL
	HULL	Ulefone Armor 9E 8/128G...	HULL	HULL	HULL	HULL	HULL	HULL	HULL	HULL
	HULL	Ulefone Armor 9F 8/128G...	HULL	HULL	HULL	HULL	HULL	HULL	HULL	HULL

The bottom right corner of the interface shows a "Result Grid" icon.

18. Brand name, Operating systems and model name of one of the dataset were inserted into another dataset with similar attributes whose brand name is OnePlus

```

use dmdd;
INSERT INTO
output(brand_name,model_name,screen_size,best_price)
SELECT Brand, Model, Screen_size, Price FROM
ndtv_data_final
WHERE Brand like
'OnePlus';SELECT *
FROM output;

```

The screenshot shows a MySQL Workbench interface. The SQL editor contains the following code:

```

1 • use dmdd;
2 • INSERT INTO output(brand_name,model_name,screen_size,best_price)
3     SELECT Brand, Model, Screen_size, Price FROM ndtv_data_final
4     WHERE Brand like 'OnePlus';
5 • SELECT * FROM output;

```

The Result Grid displays the following data:

	Id	brand_name	model_name	os	popularity	best_price	lowest_price	highest_price	sellers_amount
	1212	Apple	iPhone XR Dual Sim 128GB Black (MT192)	iOS	765	22520	18399	24855	15
	1213	Apple	iPhone XR Dual Sim 64GB Black (MT122)	iOS	695	20143	16862	22849	26
	1214	Apple	iPhone XS 256GB Gold (MT9K2)	iOS	1126	21463	15281	25617	68
	1215	Apple	iPhone XS 512GB Silver (MT9M2)	iOS	803	22842	17832	27500	62
	1216	Apple	iPhone XS 64GB Space Gray (MT9E2)	iOS	1187	19790	12505	23928	53
	1217	Apple	iPhone XS Max 256GB Gold (MT552)	iOS	1128	24184	18399	30600	37
	1218	Apple	iPhone XS Max 512GB Space Gray (MT622)	iOS	842	27190	21150	30200	47
	1219	Apple	iPhone XS Max 64GB Gold (MT522)	iOS	1101	22685	16018	27900	61
	1220	Apple	iPhone XS Max Dual Sim 64GB Gold (MT732)	iOS	530	24600	21939	33720	28
	1221	HUAWEI	nova ST 6/128GB Black (51094MEU)	Android	1174	8804	7999	9999	18
	1222	ZTE	nubia Red Magic 5G 8/128GB Black	Android	752	18755	18500	19010	2
	<i>NULL</i>	OnePlus	7T	<i>NULL</i>	<i>NULL</i>	34930	<i>NULL</i>	<i>NULL</i>	<i>NULL</i>
	<i>NULL</i>	OnePlus	7T Pro	<i>NULL</i>	<i>NULL</i>	52990	<i>NULL</i>	<i>NULL</i>	<i>NULL</i>
	<i>NULL</i>	OnePlus	7 Pro	<i>NULL</i>	<i>NULL</i>	39995	<i>NULL</i>	<i>NULL</i>	<i>NULL</i>
	<i>NULL</i>	OnePlus	6T	<i>NULL</i>	<i>NULL</i>	31999	<i>NULL</i>	<i>NULL</i>	<i>NULL</i>
	<i>NULL</i>	OnePlus	6	<i>NULL</i>	<i>NULL</i>	34999	<i>NULL</i>	<i>NULL</i>	<i>NULL</i>
	<i>NULL</i>	OnePlus	3T	<i>NULL</i>	<i>NULL</i>	19999	<i>NULL</i>	<i>NULL</i>	<i>NULL</i>
	<i>NULL</i>	OnePlus	3	<i>NULL</i>	<i>NULL</i>	22999	<i>NULL</i>	<i>NULL</i>	<i>NULL</i>
	<i>NULL</i>	OnePlus	7	<i>NULL</i>	<i>NULL</i>	29999	<i>NULL</i>	<i>NULL</i>	<i>NULL</i>
	<i>NULL</i>	OnePlus	5T	<i>NULL</i>	<i>NULL</i>	28499	<i>NULL</i>	<i>NULL</i>	<i>NULL</i>
	<i>NULL</i>	OnePlus	5	<i>NULL</i>	<i>NULL</i>	17499	<i>NULL</i>	<i>NULL</i>	<i>NULL</i>
	<i>NULL</i>	<i>OnePlus</i>	<i>~ ~</i>	<i>NULL</i>	<i>NULL</i>	<i>~ ~</i>	<i>NULL</i>	<i>NULL</i>	<i>NULL</i>

Output 1 × Read Only

19. Which of the following mobile phones have following screen size, where the quantity is above 10?

use dmdd;

```

SELECT Screen_size, count(Id) AS
No_of_PhonesFROM ndtv_data_final
GROUP BY
Screen_size
HAVING
count(Id)>10;

```

The screenshot shows a SQL query editor interface. At the top, there are tabs for various SQL files. Below the tabs is a toolbar with icons for file operations, search, and other utilities. A dropdown menu says "Limit to 2000 rows". The main area contains a query window with the following code:

```
1 • use dmdd;
2 • SELECT Screen_size, count(Id) AS No_of_Phones
3   FROM ndtv_data_final
4   GROUP BY Screen_size
5   HAVING count(Id) > 10;
```

Below the code is a result grid table with two columns: "Screen_size" and "No_of_Phones". The data is as follows:

Screen_size	No_of_Phones
6.4	21
6.3	28
5.5	251
6	53
5.99	14
6.2	23
4.7	37
5.7	51
5.2	82
5	404
4	71
4.5	80
6.22	15
5.45	25

On the right side of the interface, there is a vertical sidebar with icons for different features: Result Grid (selected), Form Editor, Field Types, Query Stats, and Execution Plan. At the bottom left, it says "Result 21" and at the bottom right, it says "Read Only".

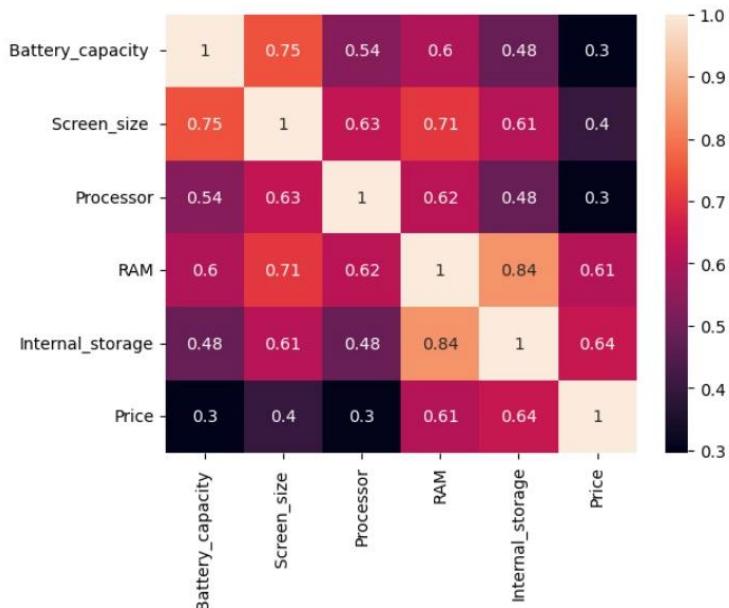
20. Self joined brands from both the tables

```
use dmdd;
SELECT ndtv_data_final.Brand,
       output.brand_name
FROM ndtv_data_final, output
WHERE ndtv_data_final.Id <> output.Id
```

Analyzing and visualizing the data:

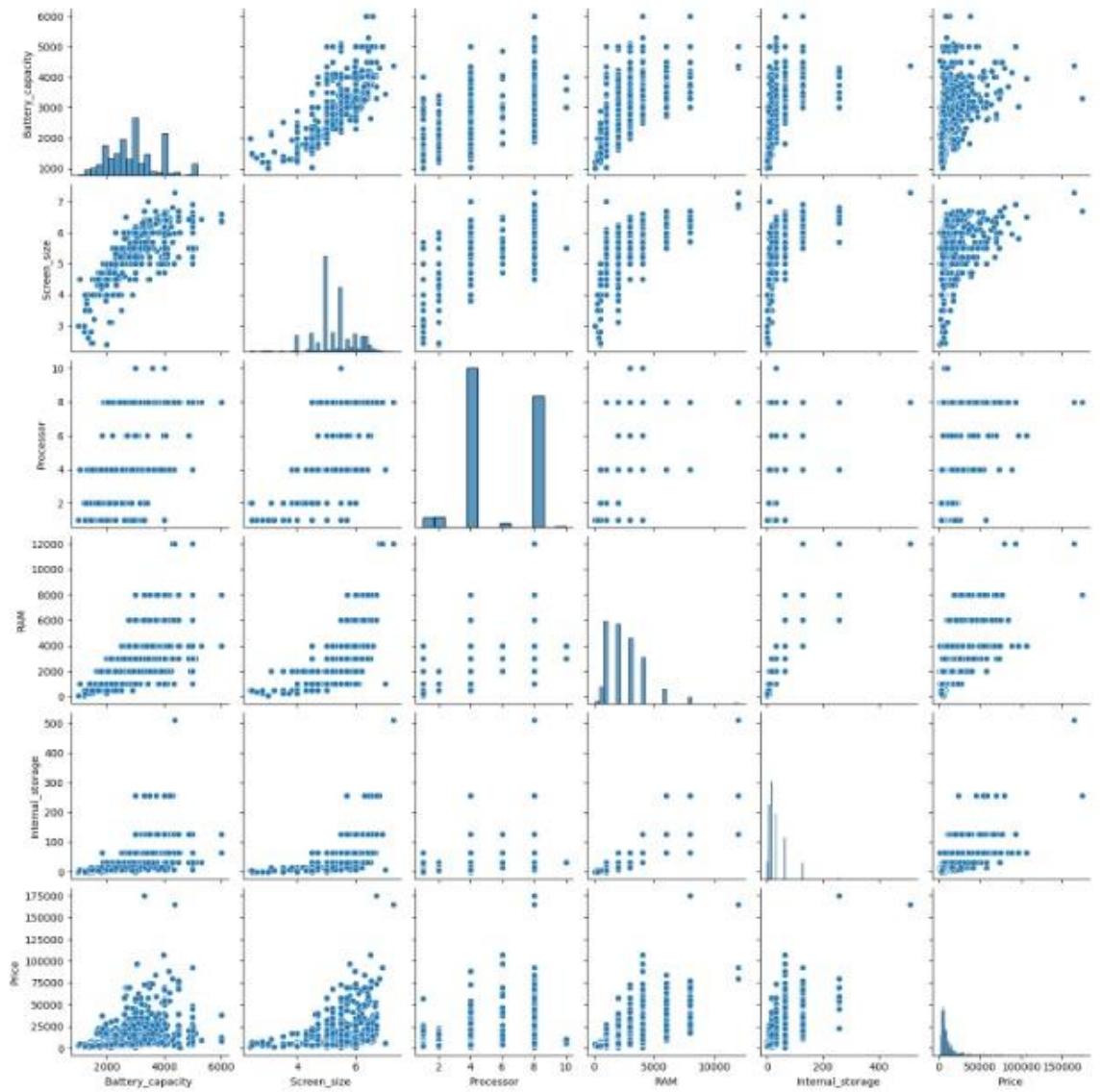
```
In [24]: #Producing heatmap of the specific attributes of mobile data
sns.heatmap(corelation,xticklabels=corelation.columns, yticklabels=corelation.columns, annot=True)

Out[24]: <AxesSubplot:>
```



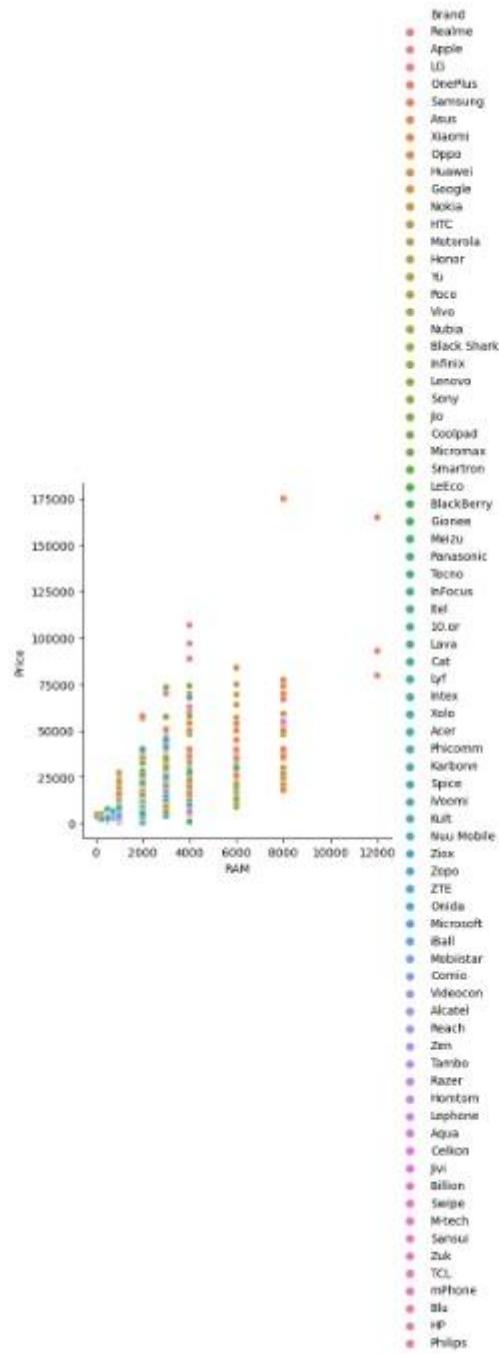
```
In [25]: #Pairplot of each attribute with another  
sns.pairplot(mobile)
```

```
Out[25]: <seaborn.axisgrid.PairGrid at 0x22e77def190>
```



```
In [11]: #Relational plot of RAM and Price sorted according to the Brand  
sns.relplot(x= 'RAM', y='Price', hue='Brand', data=mobile)
```

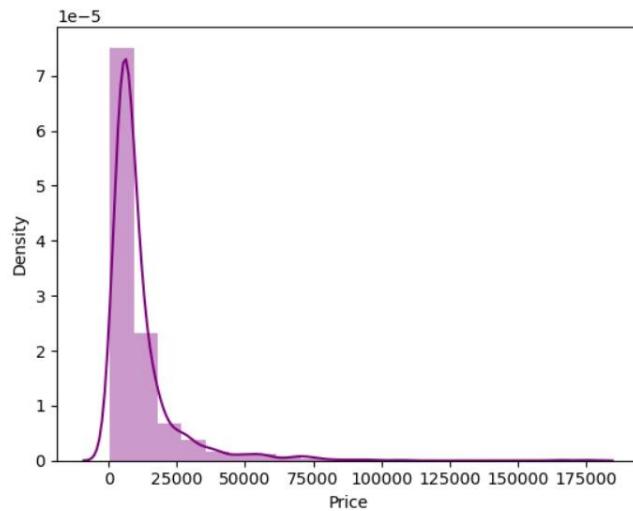
```
Out[11]: <seaborn.axisgrid.FacetGrid at 0x22e78c1adc80>
```



```
In [49]: #Distplot of Price vs Density
sns.distplot(mobile['Price'], bins=20, color='purple')
```

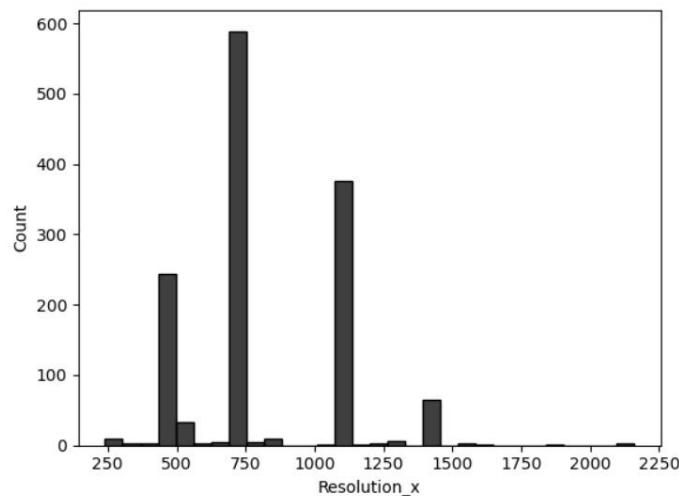
C:\Users\Harsh\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
warnings.warn(msg, FutureWarning)

```
Out[49]: <AxesSubplot:xlabel='Price', ylabel='Density'>
```

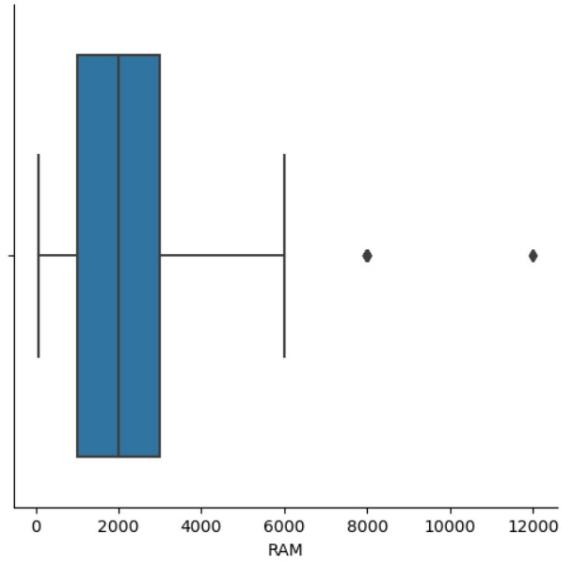


```
In [52]: #Histplot of Price vs Count
sns.histplot(data['Resolution_x'], bins=30, color='black')
```

```
Out[52]: <AxesSubplot:xlabel='Resolution_x', ylabel='Count'>
```



```
In [61]: #Catplot of RAM attribute  
sns.catplot(x='RAM', kind= 'box', data=mobile)  
Out[61]: <seaborn.axisgrid.FacetGrid at 0x22e0210e5e0>
```



Normalization of Data:

Views:

Q1. Which brand have what quantity of phones?

use dmdd;

CREATE VIEW group_by AS

```
SELECT Brand,COUNT(Name) AS No_of_Phones  
FROM ndtv_data_final  
GROUP BY Brand;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

group_by 1 × Read Only

Brand	No_of_Phones
Realme	18
Apple	17
LG	42
OnePlus	11
Samsung	101
Asus	37
Xiaomi	47
Oppo	35
Huawei	12
Google	10
Nokia	35
HTC	35
Motorola	42
Honor	33
Yu	13
Poco	2
Vivo	52
Nubia	12
Black Sh...	1
Infinix	17
Lenovo	42
Sony	28
Jio	1
Coolpad	19

Q2. Which model of the smartphone has what clockspeed?

use dmdd;

CREATE VIEW right_join AS

SELECT Model,clock_speed

FROM ndtv_data_final

RIGHT JOIN test

ON ndtv_data_final.Id = test.Id;

The screenshot shows a DBeaver interface with a query editor at the top containing the SQL command:

```
1 •  SELECT * FROM dmdd.right_join;
```

Below the query is a result grid titled "Result Grid". The grid has two columns: "Model" and "clock_speed". The data is as follows:

Model	clock_speed
X2 Pro	1.8
iPhone 11 Pro Max	0.5
iPhone 11	2.8
G8X ThinQ	0.5
7T	1.4
7T Pro	2.9
Galaxy Note 10+	2.4
ROG Phone 2	2.4
Redmi K20 Pro	2.9
K3	0.5
X	2.2
Redmi K20	2.9
7 Pro	1.4
Reno 10x Zoom	2.2
3 Pro	1.8
P30 Pro	1
Redmi Note 7 Pro	0.5
Mate 20 Pro	2.9
V40 ThinQ	1.7
6T	2.6
iPhone XR	0.6
iPhone XS Max	2.6
iPhone XS	1.7
Pixel 3 XL	0.7

The interface includes a toolbar with various icons, a "Result Grid" tab selected in the top bar, and a sidebar on the right with icons for Result Grid, Form Editor, Field Types, Query Stats, and Execution Plan. A "Read Only" status is indicated at the bottom right.

Q3. Which brand has what release dates of its phones? (cross join)

use dmdd;

CREATE VIEW cross_join AS

SELECT ndtv_data_final.Brand, output.release_date AS Release_Date

FROM output

CROSS JOIN dmdd.ndtv_data_final;

Q4. Which smartphone has what weight, pixel height and pixel width?

use dmdd;

CREATE VIEW left_join AS

```
SELECT Name,mobile_wt AS Mobile_Weight,px_height As Pixel_Height,px_width  
AS Pixel_Width
```

FROM ndtv_data_final

LEFT JOIN test

```
ON ndtv_data_final.Id = test.Id;
```

The screenshot shows a database interface with a query editor at the top containing the SQL command: `SELECT * FROM dmdd.left_join;`. Below the editor is a "Result Grid" displaying a table with four columns: Name, Mobile_Weight, Pixel_Height, and Pixel_Width. The table lists various smartphone models with their respective specifications. On the right side of the interface, there is a vertical toolbar with icons for different features: Result Grid (selected), Form Editor, Field Types, Query Stats, and Execution Plan. A status bar at the bottom indicates "left_join 1" and "Read Only".

Name	Mobile_Weight	Pixel_Height	Pixel_Width
Realme X2 Pro	193	226	1412
iPhone 11 Pro Max	191	746	857
iPhone 11	186	1270	1366
LG G8X ThinQ	96	295	1752
OnePlus 7T	108	749	810
OnePlus 7T Pro	198	569	939
Samsung Galaxy Note 10+	156	1283	1374
Asus ROG Phone 2	111	1312	1880
Xiaomi Redmi K20 Pro	101	556	876
Oppo K3	171	52	1009
Realme X	80	503	1336
Xiaomi Redmi K20	171	235	1671
OnePlus 7 Pro	87	829	1893
Oppo Reno 10x Zoom	158	227	1856
Realme 3 Pro	193	1315	1323
Huawei P30 Pro	106	185	1832
Redmi Note 7 Pro	163	613	1511
Huawei Mate 20 Pro	186	335	532
LG V40 ThinQ	194	82	1771
OnePlus 6T	189	47	559
Apple iPhone XR	151	504	1930
Apple iPhone XS Max	101	521	1591
Apple iPhone XS	115	777	1587
Google Pixel 3 XL	97	512	1111

Q5. Which phone model has what release date? (inner join)

```
use dmdd;
```

```
CREATE VIEW inner_join AS
```

```
SELECT Model,release_date
```

```
FROM ndtv_data_final
```

```
INNER JOIN output
```

```
ON ndtv_data_final.Id= output.Id;
```

The screenshot shows a MySQL Workbench interface with a query editor window. The title bar has tabs for multiple files, and the active tab is 'inner_join'. The query in the editor is:

```
1 •  SELECT * FROM dmdd.inner_join;
```

The results grid displays data from the 'inner_join' table with two columns: 'Model' and 'release_date'. The data includes various smartphone models and their release dates. A vertical toolbar on the right side of the results grid provides options for Result Grid, Form Editor, Field Types, Query Stats, and Execution Plan.

Model	release_date
X2 Pro	20-Sep
iPhone 11 Pro Max	20-Sep
iPhone 11	20-Sep
7T	18-Jun
Galaxy Note 10+	19-Jan
ROG Phone 2	21-Jan
Redmi K20 Pro	18-Dec
K3	18-Dec
X	18-Dec
P20 Pro	18-May
6	18-May
ZenFone 5Z (ZS6...	18-May
iPhone 8 Plus	18-Nov
iPhone X	18-Nov
Moto Z2 Force	20-Mar
U11	20-Nov
8	18-Sep
Galaxy S8	20-Sep
G6	20-Oct
3T	20-Oct
Galaxy C9 Pro	20-Oct
iPhone 7	20-Sep
Galaxy A8	19-Sep
X2	20-Sep

Q6. What is the total number of brands?

```
use dmdd;
```

```
CREATE VIEW union_tables AS
```

```
SELECT Brand FROM ndtv_data_final
```

```
UNION
```

```
SELECT brand_name FROM output;
```

The screenshot shows a MySQL Workbench interface. At the top, there's a toolbar with various icons. Below it, a query window displays the following SQL command:

```
1 •   SELECT * FROM dmdd.union_tables;
```

The results are presented in a "Result Grid" table. The table has one column labeled "Brand". The data contains a list of smartphone brands, including Realme, Apple, LG, OnePlus, Samsung, Asus, Xiaomi, Oppo, Huawei, Google, Nokia, HTC, Motorola, Honor, Yu, Poco, Vivo, Nubia, Black Sh..., Infinix, Lenovo, Sony, Jio, and Coolpad. The table has a header row and approximately 30 data rows.

At the bottom of the results window, it says "union_tables 1" and "Read Only". To the right of the results grid, there's a vertical toolbar with icons for "Result Grid", "Form Editor", "Field Types", "Query Stats", and "Execution Plan".

Q7. What is the operating system for what number of phones?

```
use dmdd;  
CREATE VIEW having_count AS  
SELECT Operating_system, count(Id) AS No_of_Phones  
FROM ndtv_data_final  
GROUP BY Operating_system  
HAVING count(Id) > 5;
```

The screenshot shows a MySQL Workbench interface. At the top, there are tabs for various SQL files (SQL File 19*, SQL File 20*, etc.) and a current tab for 'having_count'. Below the tabs is a toolbar with icons for file operations, search, and other database functions. A status bar at the bottom indicates 'Limit to 2000 rows'.

In the main area, a query is run:

```
1 •  SELECT * FROM dmdd.having_count;
```

The result grid displays the following data:

Operating_system	No_of_Phones
Android	1298
iOS	17
Cyanogen	10
BlackBerry	10
Windows	19

On the right side, there is a vertical toolbar with icons for different features: Result Grid (selected), Form Editor, Field Types, Query Stats, and Execution Plan. At the bottom right, it says 'Read Only'.

Q8. What is the total number of phones from both the dataset?

```
use dmdd;
```

```
CREATE VIEW union_all AS
```

```
SELECT Model From ndtv_data_final
```

```
UNION ALL
```

```
SELECT model_name FROM output;
```

The screenshot shows a MySQL Workbench interface. The query editor window contains the following SQL code:

```
1 •  SELECT * FROM dmdd.union_all;
```

The results grid displays a list of smartphone models, each preceded by a right-pointing arrow icon:

Model
X2 Pro
iPhone 11 Pro Max
iPhone 11
G8X ThinQ
7T
7T Pro
Galaxy Note 10+
ROG Phone 2
Redmi K20 Pro
K3
X
Redmi K20
7 Pro
Reno 10x Zoom
3 Pro
P30 Pro
Redmi Note 7 Pro
Mate 20 Pro
V40 ThinQ
6T
iPhone XR
iPhone XS Max
iPhone XS
Pixel 3 XL
union_all 1

The interface includes various toolbars and a sidebar with icons for Result Grid, Form Editor, Field Types, Query Stats, and Execution Plan. A "Read Only" status indicator is visible at the bottom right.

Q9. What is the average price of all the smartphones?

use dmdd;

CREATE VIEW avg_join AS

SELECT AVG(ndtv_data_final.Price) AS Average_Price

FROM ndtv_data_final

INNER JOIN output ON ndtv_data_final.Id = output.Id;

The screenshot shows a MySQL Workbench interface. At the top, there's a toolbar with various icons. Below it, a query window displays the SQL command: `SELECT * FROM dmdd.avg_join;`. The main area shows the results of this query in a grid. The grid has one row and two columns. The first column is labeled "Average_Price" and contains the value "11316.2311". To the right of the grid is a vertical sidebar with several tabs: "Result Grid" (which is selected and highlighted in blue), "Form Editor", "Field Types", "Query Stats", and "Execution Plan". At the bottom of the interface, there's a status bar showing "avg_join 1" and a "Read Only" indicator.

Average_Price	
11316.2311	

Q10. Which brands have what number of phones with internal storage more than 128GB?

```
use dmdd;
```

```
CREATE VIEW where_group_by AS
```

```
SELECT Brand, count(Internal_storage) AS Phones_with_higher_storage
```

```
FROM ndtv_data_final
```

```
WHERE Internal_storage >= 128
```

GROUP BY Brand;

```
1 •  SELECT * FROM dddd.where_group_by;
```

The screenshot shows a database query results interface. At the top, there is a command line with the query: "SELECT * FROM dddd.where_group_by;". Below the command line is a "Result Grid" table. The table has two columns: "Brand" and "Phones_with_higher_storage". The data in the table is as follows:

Brand	Phones_with_higher_storage
LG	4
OnePlus	5
Samsung	16
Asus	4
Xiaomi	2
Realme	1
Oppo	11
Huawei	6
HTC	2
Vivo	9
Nubia	3
Motorola	2
Black Sh...	1
Honor	6
Nokia	1
Infinix	1
Micromax	1
Meizu	1
Panasonic	1
Tecno	1

On the right side of the interface, there is a vertical toolbar with icons and labels: "Result Grid" (selected), "Form Editor", "Field Types", "Query Stats", and "Execution Plan". At the bottom left, it says "where_group_by 1 ×" and at the bottom right, it says "Read Only".

Q11. Which phones have what processor count and core capacity?

use dddd;

CREATE VIEW left_outer_join AS

SELECT model,pc AS Processor_Count, n_cores AS Core_Capacity

FROM ndtv_data_final

LEFT OUTER JOIN test

```
ON ndtv_data_final.Id = test.Id
```

The screenshot shows a MySQL Workbench interface with a query editor window. The query is:

```
1 • SELECT * FROM dmdd.left_outer_join;
```

The results are displayed in a grid:

model	Processor_Count	Core_Capacity
X2 Pro	16	3
iPhone 11 Pro Max	12	5
iPhone 11	4	3
G8X ThinQ	20	8
7T	18	6
7T Pro	9	8
Galaxy Note 10+	3	2
ROG Phone 2	2	1
Redmi K20 Pro	19	5
K3	20	3
X	6	7
Redmi K20	6	3
7 Pro	3	2
Reno 10x Zoom	15	5
3 Pro	9	8
P30 Pro	7	8
Redmi Note 7 Pro	17	2
Mate 20 Pro	15	4
V40 ThinQ	3	2
6T	20	4
iPhone XR	17	1
iPhone XS Max	5	5
iPhone XS	10	2
Pixel 3 XL	18	5

The sidebar on the right includes icons for Result Grid, Form Editor, Field Types, Query Stats, and Execution Plan. A note at the bottom right says "Read Only".

Q12. What is the highest and lowest price of the phones of each brand?

```
use dmdd;
```

```
CREATE VIEW right_outer_join AS
SELECT output.Id, Name, Brand, highest_price, lowest_price
FROM ndtv_data_final
RIGHT OUTER JOIN output
ON ndtv_data_final.Id = output.Id
```

1 • `SELECT * FROM dmdd.right_outer_join;`

Result Grid

	Id	Name	Brand	highest_price	lowest_price
▶	1	Realme X2 Pro	Realme	2489	1659
	2	iPhone 11 Pro Max	Apple	2489	1659
	3	iPhone 11	Apple	2489	1659
	5	OnePlus 7T	OnePlus	11099	10631
	7	Samsung Galaxy Note 10+	Samsung	5295	4733
	8	Asus ROG Phone 2	Asus	5222	4990
	9	Xiaomi Redmi K20 Pro	Xiaomi	5372	4646
	10	Oppo K3	Oppo	5372	4646
	11	Realme X	Realme	5559	4897
	30	Huawei P20 Pro	Huawei	5448	4970
	31	OnePlus 6	OnePlus	5448	4970
	32	Asus ZenFone 5Z (ZS620KL)	Asus	5448	4970
	40	Apple iPhone 8 Plus	Apple	6526	5290
	41	Apple iPhone X	Apple	5759	4599
	42	Motorola Moto Z2 Force	Motorola	5699	5599
	43	HTC U11	HTC	7786	6840
	44	Nokia 8	Nokia	8196	7049
	46	Samsung Galaxy S8	Samsung	3185	2168
	47	LG G6	LG	3299	2889
	48	OnePlus 3T	OnePlus	3899	3581
	49	Samsung Galaxy C9 Pro	Samsung	3899	3581
	52	Apple iPhone 7	Apple	3299	2999
	63	Samsung Galaxy A8	Samsung	2689	2395
	68	Poco X2	Poco	4049	3550

it_outer_join 1 × Read Only

Q13. What is the name, popularity and battery power for each mobile phone where popularity is greater than 500?

use dmdd;

CREATE VIEW triple_join AS

SELECT Name,popularity,battery_power

FROM ndtv_data_final

JOIN output

ON ndtv_data_final.Id = output.Id

JOIN test

ON output.Id = test.Id

WHERE popularity>500;

The screenshot shows a database interface with a query editor at the top containing the SQL code:

```
1 • SELECT * FROM dmdd.triple_join;
```

The results are displayed in a "Result Grid" table below:

Name	popularity	battery_power
Apple iPhone 8 Plus	1111	567
Apple iPhone X	1144	1952
Motorola Moto Z2 Force	663	822
HTC U11	900	685
Nokia 8	1063	1388
Samsung Galaxy S8	616	1411
LG G6	629	1094
OnePlus 3T	790	1653
Samsung Galaxy C9 Pro	793	916
Apple iPhone 7	1031	632
Samsung Galaxy A8	883	578
Poco X2	1132	1206
Samsung Galaxy Note ...	921	1366
Realme XT	1156	1391
Redmi Note 8 Pro	701	979
Realme 5 Pro	1139	1999
Asus 6Z	1118	1982
Samsung Galaxy M40	1117	1373
Redmi Note 7S	620	1151
Motorola One Vision	798	1650
Samsung Galaxy A70	953	1465
Black Shark 2	661	1809
OnePlus 7	989	757
LG G8s ThinQ	1042	1034

A sidebar on the right contains navigation links: Result Grid (selected), Form Editor, Field Types, Query Stats, and Execution plan. A note at the bottom right says "Read Only".

Q14. Which phones have the mobile weight less than 100?

```
use dmdd;  
CREATE VIEW where_exists AS  
SELECT Id,Name  
FROM ndtv_data_final  
WHERE EXISTS (SELECT mobile_wt FROM test WHERE ndtv_data_final.Id = test.Id  
AND mobile_wt < 100);
```

The screenshot shows a database interface with a toolbar at the top containing various icons. A dropdown menu says "Limit to 2000 rows". Below the toolbar, a query window displays the SQL command: "SELECT * FROM dmdd.where_exists;". The main area is a "Result Grid" showing a list of products with columns "Id" and "Name". The grid contains approximately 147 rows. The right side of the interface has a vertical sidebar with icons for "Result Grid" (selected), "Form Editor", "Field Types", "Query Stats", and "Execution Plan". At the bottom right of the grid area, there is a "Read Only" indicator.

Id	Name
4	LG G8X ThinQ
11	Realme X
13	OnePlus 7 Pro
24	Google Pixel 3 XL
28	LG G7+ ThinQ
30	Huawei P20 Pro
37	HTC U11+
40	Apple iPhone 8 Plus
46	Samsung Galaxy S8
60	Google Nexus 6P
61	Apple iPhone 6s Plus
66	Xiaomi Redmi Note
89	Asus 6Z
90	Samsung Galaxy M40
92	Motorola One Vision
96	Samsung Galaxy A70
100	Vivo V15 Pro
103	Samsung Galaxy M20
108	Realme U1
123	Nokia 6.1 Plus
127	Realme 1
136	Infinix Zero 5 Pro
139	Honor 9i
147	Goole Pixel 2 XL

Q15. The brand name APPLE has what number of sellers?

use dmdd;

CREATE VIEW where_less_than AS

SELECT ndtv_data_final.Id, Brand, sellers_amount AS No_of_sellers

FROM ndtv_data_final

JOIN output

ON ndtv_data_final.Id = output.Id

where ndtv_data_final.Brand='Apple' and output.sellers_amount<20;

The screenshot shows a MySQL Workbench interface. At the top, there's a toolbar with various icons. Below it is a query editor window containing the SQL command: `1 • SELECT * FROM dmdd.where_less_than;`. The main area displays a result grid titled "Result Grid". The grid has three columns: "Id", "Brand", and "No_of_sellers". The data shows four rows for the brand "Apple":

	Id	Brand	No_of_sellers
▶	40	Apple	11
	41	Apple	19
	52	Apple	17
	192	Apple	10

To the right of the result grid is a vertical toolbar with several icons:

- Result Grid (selected)
- Form Editor
- Field Types
- Query Stats
- Execution Plan

At the bottom of the interface, there's a status bar with the text "e_less_than 1" and a "Read Only" indicator.

Q16. Which of the following mobile phones are black in colour with their prices above 40000?

use dmdd;

CREATE VIEW where_like AS

SELECT model_name, Price

FROM ndtv_data_final

JOIN output

ON ndtv_data_final.Id = output.Id

```
where output.model_name like '%Black%' and ndtv_data_final.Price>40000;
```

The screenshot shows a database query results window. At the top, there are various toolbar icons. Below the toolbar, a SQL query is displayed: `SELECT * FROM dddd.where_like;`. The main area contains a "Result Grid" table with two columns: "model_name" and "Price". The table lists numerous smartphone models along with their prices. A vertical sidebar on the right provides navigation links for "Result Grid", "Form Editor", "Field Types", "Query Stats", and "Execution Plan". The bottom of the window shows the query name "where_like 1" and a "Read Only" status indicator.

model_name	Price
1 5033D 1/16GB Volcano Black (5033D-2LALUAF)	106900
1 5033D 1/16GB Volcano Black (5033D-2LALUAF)	62900
10 Lite 4/64GB Black	79699
15 4/64GB Black	49990
16 6/128GB Black	47000
16 6/64GB Black	69999
8x Max 4/64GB Black	59090
A60 Pro 3/16GB Black	54665
BV9600 Pro 6/128GB Black	50650
BV9600 Pro 6/128GB Black	56699
BV9800 6/128Gb Black	57915
Galaxy A20e SM-A202F 3/32GB Black SM-A202F...	45999
Galaxy M10 SM-M105F 2/16GB Black (SM-M105...	42021
Mix 2 M2 4/64GB Black	92999
Pixel 32GB (Quite Black)	74990
RAZR 2019 XT2000-2 Noir Black	57900
ROG Phone 3 ZS661KS 12/512GB Black	67900
Redmi Note 9T 4/128GB Nightfall Black	44990
iPhone 7 Plus 32GB Black (MNQM2)	88719

Q17. Brand name, Operating systems and model name of one of the dataset were inserted into another dataset with similar attributes whose price is greater than 10000

```
use dddd;
```

```
INSERT INTO ndtv_data_final(Brand,Model,Operating_system)
```

```
SELECT brand_name, model_name,os FROM output
```

```
WHERE highest_price>10000;
```

```
CREATE VIEW inserted_greater_than AS
```

```
SELECT * FROM dmdd.ndtv_data_final;
```

The screenshot shows a database query results grid. The query is:

```
1 •  SELECT * FROM dmdd.inserted_greater_than;
```

The results grid displays the following columns: Id, Name, Brand, Model, Battery_capacity, Screen_size, Touchscreen, Resolution_x, Resolution_y, and Processor. The data includes various models from brands like Realme, Apple, Samsung, and OnePlus. The results grid has a header row and approximately 22 data rows.

	Id	Name	Brand	Model	Battery_capacity	Screen_size	Touchscreen	Resolution_x	Resolution_y	Processor
▶	1	Realme X2 Pro	Realme	X2 Pro	4000	6.5	Yes	1080	2400	8
	2	iPhone 11 Pro Max	Apple	iPhone 11 Pro Max	3969	6.5	Yes	1242	2688	6
	3	iPhone 11	Apple	iPhone 11	3110	6.1	Yes	828	1792	6
	4	LG G8X ThinQ	LG	G8X ThinQ	4000	6.4	Yes	1080	2340	8
	5	OnePlus 7T	OnePlus	7T	3800	6.55	Yes	1080	2400	8
	6	OnePlus 7T Pro	OnePlus	7T Pro	4085	6.67	Yes	1440	3120	8
	7	Samsung Galaxy Note 10+	Samsung	Galaxy Note 10+	4300	6.8	Yes	1440	3040	8
	8	Asus ROG Phone 2	Asus	ROG Phone 2	6000	6.59	Yes	1080	2340	8
	9	Xiaomi Redmi K20 Pro	Xiaomi	Redmi K20 Pro	4000	6.39	Yes	1080	2340	8
	10	Oppo K3	Oppo	K3	3765	6.5	Yes	1080	2340	8
	11	Realme X	Realme	X	3765	6.53	Yes	1080	2340	8
	12	Xiaomi Redmi K20	Xiaomi	Redmi K20	4000	6.39	Yes	1080	2340	8
	13	OnePlus 7 Pro	OnePlus	7 Pro	4000	6.67	Yes	1440	3120	8
	14	Oppo Reno 10x Zoom	Oppo	Reno 10x Zoom	4065	6.6	Yes	1080	2340	8
	15	Realme 3 Pro	Realme	3 Pro	4045	6.3	Yes	1080	2340	8
	16	Huawei P30 Pro	Huawei	P30 Pro	4200	6.47	Yes	1080	2340	8
	17	Redmi Note 7 Pro	Xiaomi	Redmi Note 7 Pro	4000	6.3	Yes	1080	2340	8
	18	Huawei Mate 20 Pro	Huawei	Mate 20 Pro	4200	6.39	Yes	1440	3120	8
	19	LG V40 ThinQ	LG	V40 ThinQ	3300	6.4	Yes	1440	3120	8
	20	OnePlus 6T	OnePlus	6T	3700	6.41	Yes	1080	2340	8
	21	Apple iPhone XR	Apple	iPhone XR	2942	6.1	Yes	828	1792	6
	22	Apple iPhone XS Max	Apple	iPhone XS Max	2658	6.5	Yes	1242	2688	6
	23	Apple iPhone XS	Apple	iPhone XS	2650	6.0	Yes	1125	2410	6

Q18. Brand name, Operating systems and model name of one of the dataset were inserted into another dataset with similar attributes whose brand name is Oneplus

```
use dmdd;
```

```
INSERT INTO output(brand_name,model_name,screen_size,best_price)
```

```
SELECT Brand, Model, Screen_size, Price FROM ndtv_data_final
```

```
WHERE Brand like 'OnePlus';
```

```
CREATE VIEW inserted_like AS
```

```
SELECT * FROM output;
```

```
1 •  SELECT * FROM dmdd.inserted_like;
```

The screenshot shows a database query results grid titled "Result Grid". The grid displays data from the "inserted_like" table. The columns are: Id, brand_name, model_name, os, popularity, best_price, lowest_price, highest_price, and sellers_amount. The data consists of 63 rows, each representing a different mobile phone model with its specifications and popularity metrics. The grid has a toolbar at the top with options like "Filter Rows", "Export", and "Wrap Cell Content". To the right of the grid is a sidebar with icons for "Result Grid", "Form Editor", "Field Types", "Query Stats", and "Execution Plan". At the bottom left is a tab labeled "inserted_like 1" and at the bottom right is a "Read Only" status indicator.

Id	brand_name	model_name	os	popularity	best_price	lowest_price	highest_price	sellers_amount
1	ALCATEL	1 5033D 1/16GB Volcano Black (5033D-2LALUAF)	Android	323	1803	1659	2489	36
2	ALCATEL	1 5033D 1/16GB Volcano Black (5033D-2LALUAF)	Android	299	1803	1659	2489	36
3	ALCATEL	1 5033D 1/16GB Volcano Black (5033D-2LALUAF)	Android	287	1803	1659	2489	36
5	Honor	10 6/64GB Black	Android	71	10865	10631	11099	2
7	Honor	10 Lite 4/64GB Black	Android	134	4973	4733	5295	6
8	Honor	10 lite 3/128GB Blue	Android	477	5100	4990	5222	3
9	Honor	10 lite 3/64GB Black	Android	215	4948	4646	5372	8
10	Honor	10 lite 3/64GB Black	Android	179	4948	4646	5372	8
11	Honor	10 lite 3/64GB Blue	Android	437	5165	4897	5559	7
30	Meizu	15 4/64GB Black	Android	166	5272	4970	5448	3
31	Meizu	15 4/64GB Black	Android	125	5272	4970	5448	3
32	Meizu	15 4/64GB Black	Android	116	5272	4970	5448	3
40	Meizu	16 6/128GB Black	Android	1111	5926	5290	6526	11
41	Meizu	16 6/64GB Black	Android	1144	5231	4599	5759	19
42	Meizu	16X 6/128GB Dual Purple	Android	663	5649	5599	5699	2
43	Meizu	16Xs 6/128GB Pearl White	Android	900	7229	6840	7786	11
44	Meizu	16th 6/64GB Black	Android	1063	7776	7049	8196	9
46	ALCATEL	1B 5002H Prime Black (5002H-2AALUA12)	Android	616	2408	2168	3185	31
47	ALCATEL	1SE 3/32GB Power Gray (5030D-2AALUA2)	Android	629	3249	2889	3299	38
48	ALCATEL	1SE 4/128GB Agate Green (5030E-2BALUA2)	Android	790	3877	3581	3899	35
49	ALCATEL	1SE 4/128GB Agate Green (5030E-2BALUA2)	Android	793	3877	3581	3899	35
52	Nokia	2.4 2/32GB Charcoal	Android	1031	3162	2999	3299	17
63	Nokia	2720 Flip Black (16BTSB01A10)	iOS	883	2530	2395	2689	26

Q19. Which of the following mobile phones have following screen size, where the quantity is above 10?

use dmdd;

```
CREATE VIEW group_by_having AS
```

```
SELECT Screen_size, count(Id) AS No_of_Phones
```

```
FROM ndtv_data_final
```

```
GROUP BY Screen_size
```

```
HAVING count(Id) > 10;
```

The screenshot shows a database interface with a toolbar at the top containing various icons. Below the toolbar, a query window displays the SQL command: `SELECT * FROM dmdd.group_by_having;`. The main area shows a "Result Grid" with two columns: "Screen_size" and "No_of_Phones". The data is as follows:

Screen_size	No_of_Phones
6.4	21
6.3	28
5.5	251
6	53
5.99	14
6.2	23
4.7	37
5.7	51
5.2	82
5	404
4	71
4.5	80
6.22	15
5.45	25

On the right side, there is a vertical toolbar with icons for "Result Grid", "Form Editor", "Field Types", "Query Stats", and "Execution Plan". At the bottom right, it says "Read Only".

Q20. Self joined brands from both the tables

```
use dmdd;
```

```
CREATE VIEW self_join AS
```

```
SELECT ndtv_data_final.Brand, output.brand_name
```

```
FROM ndtv_data_final, output
```

```
WHERE ndtv_data_final.Id <> output.Id
```

Normalization:

1. 1NF:

First Normal Form: Satisfies 1NF requirements

The table has a primary key attribute identified as Id

No multi-value attributes present

No repeating groups

Function:

Name: new_function

The name of the routine is parsed automatically from the DDL statement. The DDL is parsed automatically while you type.

DDL:

```
1 • CREATE DEFINER='root'@'localhost' FUNCTION `new_function`(a int) RETURNS varchar(20) CHARSET utf8
2     READS SQL DATA
3     DETERMINISTIC
4     BEGIN
5         declare result varchar(20);
6         SET global log_bin_trust_function_creators=1;
7         IF a>0 THEN
8             SET result = "table is not in 1NF";
9         ELSE
10            SET result = 'table in 1 NF';
11        END IF;
12        RETURN result;
13    END
```

Routine

Apply Revert

Procedure:

Name: new_procedure

The name of the routine is parsed automatically from the DDL statement. The DDL is parsed automatically while you type.

DDL:

```
1 • CREATE DEFINER='root'@'localhost' PROCEDURE `new_procedure`(out var int)
2 • BEGIN
3     select count(*) into var from dmdd.ndtv_data_final where
4         Model like '%,%';
5 END
```

Routine

Apply Revert

Limit to 2000 rows

```
1 • use dmdd;
2
3 • set @var = 0;
4 • call ndtv_data_final.new_procedure(@var);
5 • select @var;
6
7 • select dmdd.new_function(@var);
```

Outputs:

The screenshot displays a database management interface with three main sections:

- Result Grid:** Shows a single row with the value '@var'.
- SQL Editor:** Displays the following SQL code:

```
1 •  use dmdd;
2
3 •  set @var = 0;
4 •  call ndtv_data_final.new_procedure(@var);
5 •  select @var;
6
7 •  select dmdd.new_function(@var);
```
- Results:** Shows the output of the last query: 'dmdd.new_function(@var)' with a note 'table in 1NF'.

A vertical toolbar on the right side includes icons for Result Grid, Form Editor, Field Types, Query Stats, and Execution Plan. A status bar at the bottom indicates 'Result 10' and 'Read Only'.

2. 2NF:

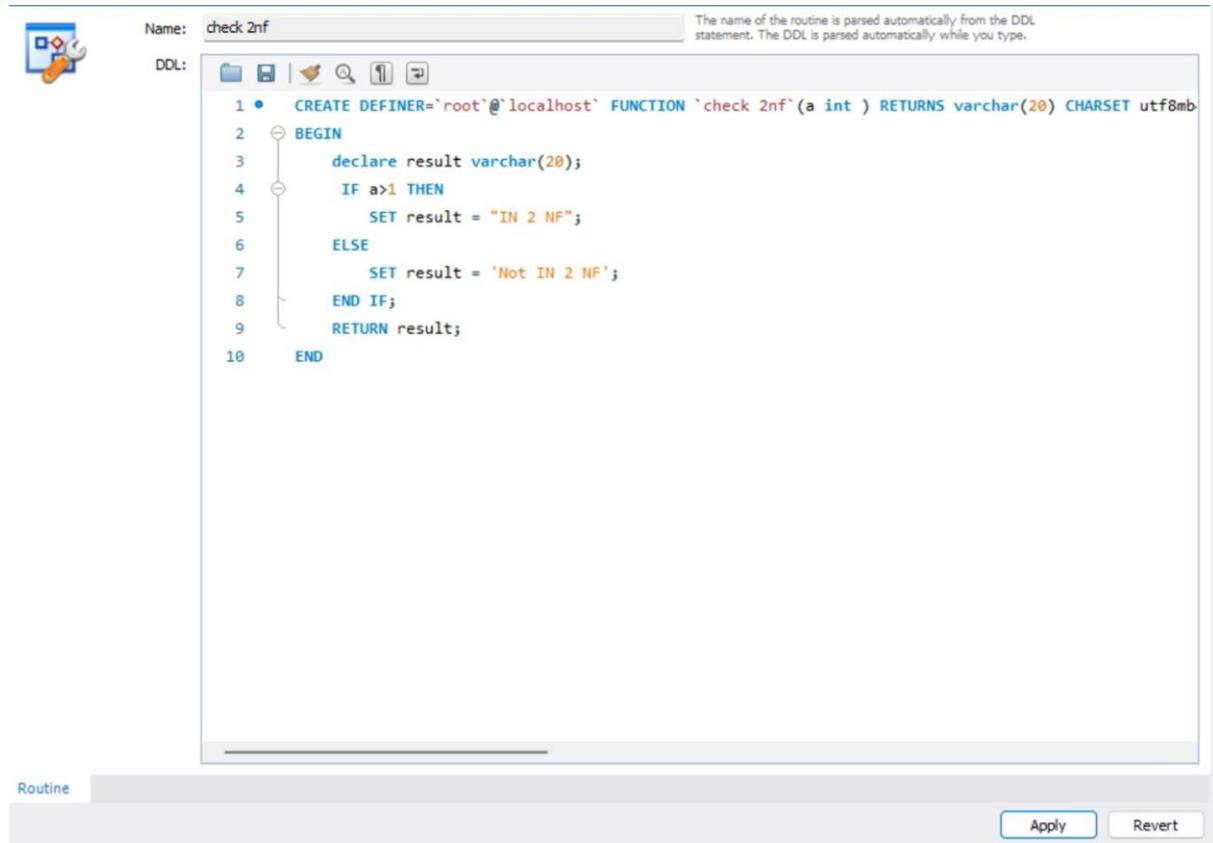
Satisfies 2NF requirements

All requirements for 1NF met

No partial dependency on any column

None of the fields have data calculated from other fields

Function:



The screenshot shows the MySQL Workbench interface with a code editor window. The window has a toolbar at the top with icons for file operations, search, and help. Below the toolbar, there are two input fields: 'Name:' containing 'check 2nf' and 'DDL:' containing the function definition. The DDL text is as follows:

```
1 • CREATE DEFINER='root'@'localhost' FUNCTION `check 2nf`(a int) RETURNS varchar(20) CHARSET utf8mb
2   BEGIN
3     declare result varchar(20);
4     IF a>1 THEN
5       SET result = "IN 2 NF";
6     ELSE
7       SET result = 'Not IN 2 NF';
8     END IF;
9     RETURN result;
10  END
```

Below the code editor, there is a tab bar with 'Routine' selected. At the bottom right of the window are 'Apply' and 'Revert' buttons.

Procedure:

The screenshot shows the MySQL Workbench interface for creating a stored procedure. The 'Routine' tab is selected. The 'Name' field contains 'new_query'. The 'DDL' pane displays the following code:

```
1 • CREATE DEFINER='root'@'localhost' PROCEDURE `new_query`(in val int ,out var int)
2 BEGIN
3     select count(distinct Brand) into var from dmdd.ndtv_data_final where Id =val;
4 END
```

The code is color-coded for syntax highlighting. A note above the DDL states: "The name of the routine is parsed automatically from the DDL statement. The DDL is parsed automatically while you type."

At the bottom right of the editor window, there are 'Apply' and 'Revert' buttons.

Name: new_query

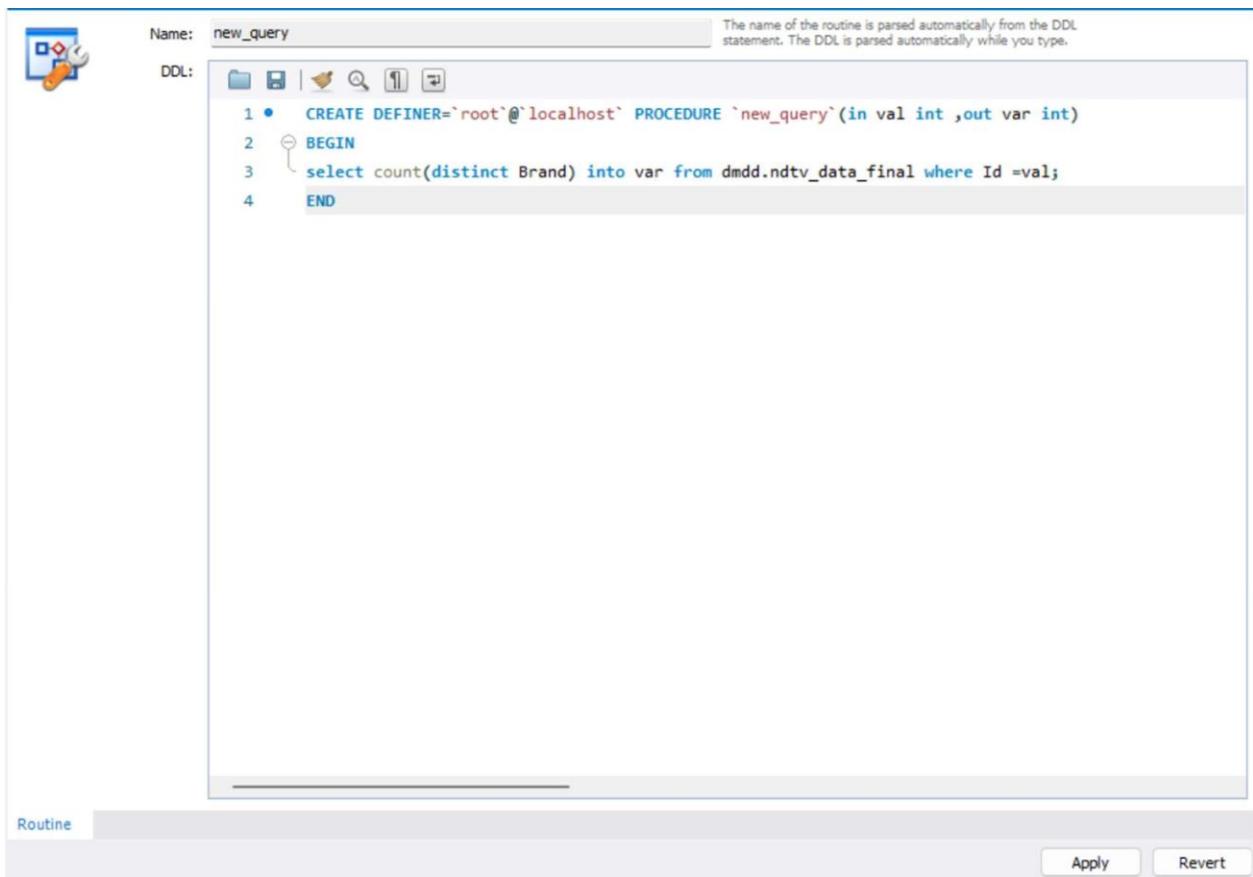
The name of the routine is parsed automatically from the DDL statement. The DDL is parsed automatically while you type.

DDL:

```
1 • CREATE DEFINER='root'@'localhost' PROCEDURE `new_query`(in val int ,out var int)
2 BEGIN
3     select count(distinct Brand) into var from dmdd.ndtv_data_final where Id =val;
4 END
```

Routine

Apply Revert



Outputs:

Result Grid | Filter Rows: _____ | Export: | Wrap Cell Content:

	@var
▶	1

The screenshot shows a database management system interface. At the top, there is a toolbar with various icons. Below the toolbar, a code editor displays the following SQL script:

```
1 •  set @var = 0;
2 •  call dmdd.new_query(6,@var);
3 •  select @var;
4
5 •  select dmdd.`check_2nf`(6);
6
7
```

Below the code editor is a results pane titled "Result Grid". It contains a single row of data:

dmdd.`check_2nf` 6)
IN 2 NF

On the right side of the interface, there is a vertical sidebar with several tabs:

- Result Grid (selected)
- Form Editor
- Field Types
- Query Stats
- Execution Plan

At the bottom left, it says "Result 29" and at the bottom right, there is a "Read Only" status indicator.

3. 3NF:

Satisfies 3NF requirements

2NF requirements satisfied

No transitive dependency

Function:

The screenshot shows the MySQL Workbench interface for creating a function. The 'Name' field is set to 'checknf3a'. The 'DDL' pane contains the following SQL code:

```
1 • CREATE DEFINER='root'@'localhost' FUNCTION `checknf3a`(a int ) RETURNS varchar(20) CHARSET utf8mb
2     READS SQL DATA
3     DETERMINISTIC
4     BEGIN
5         declare result varchar(20);
6         SET GLOBAL log_bin_trust_function_creators = 1;
7         IF a>1 THEN
8             SET result = "IN NF 3 ";
9         ELSE
10            SET result = 'not IN NF 3 ';
11        END IF;
12        RETURN result;
13    END
```

The 'Routine' tab is selected at the bottom, along with 'Apply' and 'Revert' buttons.

Procedure:

Name: new_procedure2

The name of the routine is parsed automatically from the DDL statement. The DDL is parsed automatically while you type.

DDL:

```
1 • CREATE DEFINER='root'@'localhost' PROCEDURE `new_procedure2`(in val int ,out var int)
2 • BEGIN
3     select count(distinct Brand) into var from dmdd.ndtv_data_final where Id =val;
4     END
```

Routine

Apply Revert

Limit to 2000 rows

```
1 • set @var = 0;
2 • call dmdd.new_query(6,@var);
3 • select @var;
4
5 • select dmdd.`check 2nf` (5);
6
7
```

Outputs:

Result Grid | Filter Rows: _____ | Export: | Wrap Cell Content:

	@var
▶	1

Limit to 2000 rows

```

1 •  set @var = 0;
2 •  call dmdd.new_procedure2(52, @var);
3 •  select @var;
4
5 •  select dmdd.checknf3a(5);
6

```

Result Grid | Filter Rows: _____ | Export: | Wrap Cell Content:

dmdd.checknf3a(5)
▶ IN NF 3

Result 2 x

Result Grid Form Editor Field Types Query Stats Execution Plan Read Only