

1.

```
SELECT min(EXTRACT(YEAR from o.order_purchase_timestamp)) as MinYear,  
max(EXTRACT(YEAR from o.order_purchase_timestamp)) as MaxYear,  
FROM `target-362415.7008.customers` c  
join `target-362415.7008.orders` o  
on c.customer_id = o.customer_id;
```

The screenshot displays a data analytics application interface. On the left is the 'Explorer' panel with a search bar and a tree view of data sources under 'target-362415', including '7008' and its sub-tables like 'customers', 'geolocation', 'order_items', 'order_reviews', 'orders', 'payments', 'products', and 'sellers'. The 'geolocation' table is currently selected. The main panel shows a SQL query editor with the following code:

```
1 SELECT min(EXTRACT(YEAR from o.order_purchase_timestamp)) as MinYear,  
2 max(EXTRACT(YEAR from o.order_purchase_timestamp)) as MaxYear,  
3 FROM `target-362415.7008.customers` c  
4 join `target-362415.7008.orders` o  
5 on c.customer_id = o.customer_id;  
6
```

Below the query editor, the 'Query results' section is active, showing a table with the following data:

Row	MinYear	MaxYear
1	2016	2018

The interface also includes a top navigation bar with tabs for 'RUN', 'SAVE', 'SHARE', 'SCHEDULE', and 'MORE'. A status bar at the bottom indicates 'Query completed.' and 'Press Alt+F1 for Accessibility Options.'.

```

SELECT distinct g.geolocation_city, g.geolocation_state FROM
`target-362415.7008.order_items` ordt
left join `target-362415.7008.sellers` s
on ordt.seller_id = s.seller_id
left join `target-362415.7008.geolocation` g
on s.seller_zip_code_prefix = g.geolocation_zip_code_prefix;

```

Explorer + ADD DATA

Viewing pinned projects.

- target-362415
 - External connections
 - 7008
 - customers
 - geolocation
 - order_items
 - order_reviews
 - orders
 - payments
 - products
 - sellers

Query 4 x geolocation x *Unsaved query 5 x payments x Editor 6 x *Unsaved query 7 x

Query completed.

```

1 SELECT distinct g.geolocation_city, g.geolocation_state FROM `target-362415.7008.order_items` ordt
2 left join `target-362415.7008.sellers` s
3 on ordt.seller_id = s.seller_id
4 left join `target-362415.7008.geolocation` g
5 on s.seller_zip_code_prefix = g.geolocation_zip_code_prefix;
6

```

Query results

JOB INFORMATION RESULTS JSON EXECUTION DETAILS

Row	geolocation_city	geolocation_state
1	guarulhos	SP
2	sao paulo	SP
3	são paulo	SP
4	cotia	SP
5	mogi das cruzeiros	SP
6	claudio	MG

Results per page: 50 1 - 50 of 802

PERSONAL HISTORY PROJECT HISTORY

REFRESH

2.

```
SELECT count(ordt.order_item_id) as ordercount,  
extract(year from ord.order_purchase_timestamp) as year  
FROM `target-362415.7008.order_items` ordt  
left join `target-362415.7008.orders` ord  
on ordt.order_id = ord.order_id  
group by year  
order by year asc;
```

The screenshot displays a data analytics application interface. On the left, the 'Explorer' panel shows a tree view of databases and tables, with 'payments' selected. The main area contains a SQL editor with a query that counts order items by year. Below the editor, the 'Query results' section shows a table with three rows of data. At the bottom, there are tabs for 'PERSONAL HISTORY' and 'PROJECT HISTORY', and a 'REFRESH' button.

Query results

Row	ordercount	year
1	370	2016
2	50864	2017
3	61416	2018

```

SELECT count(ordt.order_item_id) as ordercount,
extract(year from ord.order_purchase_timestamp) as year,
extract(month from ord.order_purchase_timestamp) as month
FROM `target-362415.7008.order_items` ordt
left join `target-362415.7008.orders` ord
on ordt.order_id = ord.order_id
group by year,month
order by year,month asc;

```

The screenshot displays a data analytics application interface. On the left is an 'Explorer' sidebar with a search bar and a tree view of data sources under the project 'target-362415'. The tree includes 'External connections', a folder '7008' containing 'customers', 'geolocation', 'order_items', 'order_reviews', 'orders', 'payments' (highlighted), 'products', and 'sellers'. The main area is split into a top query editor and a bottom results panel. The query editor shows a SQL query (lines 1-9) that counts order items by year and month, joined with the orders table. The results panel, titled 'Query results', shows a table with 6 rows of data. The table has columns for 'Row', 'ordercount', 'year', and 'month'. The data shows counts for years 2016 and 2017 across months 9, 10, 12, 1, 2, and 3. At the bottom right, it indicates 'Results per page: 50' and '1 - 24 of 24'.

```

1 SELECT count(ordt.order_item_id) as ordercount,
2 extract(year from ord.order_purchase_timestamp) as year,
3 extract(month from ord.order_purchase_timestamp) as month
4 FROM `target-362415.7008.order_items` ordt
5 left join `target-362415.7008.orders` ord
6 on ordt.order_id = ord.order_id
7 group by year,month
8 order by year,month asc;
9

```

Row	ordercount	year	month
1	6	2016	9
2	363	2016	10
3	1	2016	12
4	955	2017	1
5	1951	2017	2
6	3000	2017	3

```

select sum(x.ordercount),
case
when x.hour >= 6 and x.hour <12 then 'Morning'
when x.hour >= 12 and x.hour <17 then 'Afternoon'
when x.hour >= 17 and x.hour <21 then 'Dawn'
when x.hour >= 21 and x.hour <=24 then 'Night'
end as hourbin
from
(SELECT count(ordt.order_item_id) as ordercount,
extract(hour from ord.order_purchase_timestamp) as hour,
FROM `target-362415.7008.order_items` ordt
left join `target-362415.7008.orders` ord
on ordt.order_id = ord.order_id
group by hour) x
group by hourbin;

```

The screenshot displays a data analytics application interface. On the left is an 'Explorer' sidebar with a search bar and a tree view of a project named 'target-362415'. The tree includes 'External connections', a folder '7008', and several tables: 'customers', 'geolocation', 'order_items', 'order_reviews', 'orders', 'payments' (highlighted), 'products', and 'sellers'. The main workspace shows a SQL editor with a query that calculates the sum of order counts grouped by time of day (Morning, Afternoon, Dawn, Night). Below the editor, the 'Query results' section is active, showing a table with 5 rows of data. At the bottom, there are tabs for 'PERSONAL HISTORY' and 'PROJECT HISTORY', and a 'REFRESH' button.

Row	f0_	hourbin
1	36810	Afternoon
2	27151	Dawn
3	25225	Morning
4	18098	Night
5	5366	null

3.

```
select sum(x.orderitem_count) count,
x.month as month,
x.state
from
(
SELECT count(ordt.order_item_id) as orderitem_count,
extract (month from ord.order_purchase_timestamp) as Month,
g.geolocation_city as geolocation_region,
g.geolocation_state as state
FROM `target-362415.7008.order_items` ordt
left join `target-362415.7008.orders` ord
on ordt.order_id = ord.order_id
left join `target-362415.7008.sellers` s
on ordt.seller_id = s.seller_id
left join `target-362415.7008.geolocation` g
on s.seller_zip_code_prefix = g.geolocation_zip_code_prefix
group by state,geolocation_region,Month
order by state,geolocation_region,Month) x
group by x.state,x.month;
```

The screenshot displays a data analytics application interface. On the left is an 'Explorer' sidebar with a search bar and a list of projects, including 'target-362415'. The main area is titled '*Unsaved query' and contains a SQL query editor with the following code:

```
1 select sum(x.orderitem_count) count,
2 x.month as month,
3 x.state
4 from
5 (
6 SELECT count(ordt.order_item_id) as orderitem_count,
7 extract (month from ord.order_purchase_timestamp) as Month,
8 g.geolocation_city as geolocation_region,
9 g.geolocation_state as state
10 FROM `target-362415.7008.order_items` ordt
11 left join `target-362415.7008.orders` ord
12 on ordt.order_id = ord.order_id
```

Below the query editor, the 'Query results' section is active, showing a table with 4 columns: 'Row', 'count', 'month', and 'state'. The table contains 8 rows of data:

Row	count	month	state
41	116265	8	PR
42	89690	6	RJ
43	106136	8	RJ
44	52645	1	SC
45	49933	4	SC
46	201252	6	MG
47	59683	8	SC
48	51469	6	SC

At the bottom of the interface, there are tabs for 'PERSONAL HISTORY' and 'PROJECT HISTORY', and a 'REFRESH' button.

```

SELECT c.customer_state as state,
count(distinct ord.customer_id) as customer_count
FROM `target-362415.7008.orders` ord
left join `target-362415.7008.customers` c
on c.customer_id = ord.customer_id
group by c.customer_state
order by customer_count desc;

```

The screenshot shows a data analytics application interface. On the left is an 'Explorer' sidebar with a search bar and a tree view of projects. The 'target-362415' project is expanded, showing a folder '7008' which contains tables: 'customers', 'geolocation', 'order_items', 'order_reviews', 'orders', 'payments' (highlighted), 'products', and 'sellers'. The main area displays a SQL query in a text editor, with a toolbar above it containing buttons for 'RUN', 'SAVE', 'SHARE', 'SCHEDULE', and 'MORE'. A status bar at the top right indicates 'Query completed.' Below the editor, the 'Query results' section is active, showing a table with 6 rows and 3 columns: 'Row', 'state', and 'customer_c...'. The table data is as follows:

Row	state	customer_c...
1	RJ	12852
2	RS	5466
3	SP	41746
4	DF	2140
5	PR	5045
6	MT	907

Below the table, there are tabs for 'JOB INFORMATION', 'RESULTS' (selected), 'JSON', and 'EXECUTION DETAILS'. At the bottom, there are sections for 'PERSONAL HISTORY' and 'PROJECT HISTORY', and a 'REFRESH' button.

4.

```
with cte2017 as (  
SELECT sum(ordt.freight_value) as cost_17,  
FROM `target-362415.7008.orders` ord  
join `target-362415.7008.order_items` ordt  
on ord.order_id=ordt.order_id  
where extract (year from order_purchase_timestamp) = 2017),
```

```
cte2018 as (  
SELECT sum(ordt.freight_value) as cost_18,  
FROM `target-362415.7008.orders` ord  
join `target-362415.7008.order_items` ordt  
on ord.order_id=ordt.order_id  
where extract (year from order_purchase_timestamp) = 2018)
```

```
SELECT  
round(((cte2018.cost_18 - cte2017.cost_17)/cte2017.cost_17)*100,2) as  
percentage_change  
FROM cte2018, cte2017;
```

The screenshot shows a SQL IDE interface. On the left is an Explorer pane with a tree view of a project named 'target-362415'. Under 'External connections', there is a folder '7008' containing tables: 'customers', 'geolocation', 'order_items' (selected), 'order_reviews', 'orders', 'payments', 'products', and 'sellers'. The main editor displays a SQL query with line numbers 1 through 17. The query defines two CTEs, cte2017 and cte2018, and then calculates the percentage change in freight cost from 2017 to 2018. The query is executed, and a 'Query results' pane at the bottom shows a table with one row and one column, 'percentage_change', with a value of 27.44. The interface also includes a top bar with tabs for 'payments', 'Editor 6', and several 'Unsaved query' tabs, and a bottom bar with 'PERSONAL HISTORY' and 'PROJECT HISTORY' sections.

```
1 with cte2017 as (  
2   SELECT sum(ordt.freight_value) as cost_17,  
3   FROM `target-362415.7008.orders` ord  
4   join `target-362415.7008.order_items` ordt  
5   on ord.order_id=ordt.order_id  
6   where extract (year from order_purchase_timestamp) = 2017),  
7  
8   cte2018 as (  
9     SELECT sum(ordt.freight_value) as cost_18,  
10    FROM `target-362415.7008.orders` ord  
11    join `target-362415.7008.order_items` ordt  
12    on ord.order_id=ordt.order_id  
13    where extract (year from order_purchase_timestamp) = 2018)  
14  
15 SELECT  
16   round(((cte2018.cost_18 - cte2017.cost_17)/cte2017.cost_17)*100,2) as percentage_change  
17 FROM cte2018, cte2017;
```

Row	percentage_change
1	27.44


```

select sum(x.averageprice) as avgprice,
sum(x.totalprice) as total,
sum(x.frightvalue) as frieght,
x.customerstate as state
from
(select round(sum(x.price),2) as totalprice,
round(avg(x.price),2) as averageprice,
x.freightvalue as frightvalue,
x.customerstate
from
(SELECT ordt.price as price,
ordt.freight_value as freightvalue,
c.customer_state as customerstate
FROM `target-362415.7008.order_items` ordt
left join `target-362415.7008.orders` ord
on ordt.order_id = ord.order_id
left join `target-362415.7008.customers` c
on c.customer_id = ord.customer_id) x
group by x.freightvalue,x.customerstate) x
group by x.customerstate
order by avgprice desc;

```

SANDBOX Set up billing to upgrade to the full BigQuery experience. [Learn more](#) DISMISS UPGRADE

Explorer + ADD DATA IK

Q Type to search

Viewing pinned projects. Expand node

target-362415

Query Editor Q *Unsaved query + 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503 504 505 506 507 508 509 510 511 512 513 514 515 516 517 518 519 520 521 522 523 524 525 526 527 528 529 530 531 532 533 534 535 536 537 538 539 540 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555 556 557 558 559 560 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575 576 577 578 579 580 581 582 583 584 585 586 587 588 589 590 591 592 593 594 595 596 597 598 599 600 601 602 603 604 605 606 607 608 609 610 611 612 613 614 615 616 617 618 619 620 621 622 623 624 625 626 627 628 629 630 631 632 633 634 635 636 637 638 639 640 641 642 643 644 645 646 647 648 649 650 651 652 653 654 655 656 657 658 659 660 661 662 663 664 665 666 667 668 669 670 671 672 673 674 675 676 677 678 679 680 681 682 683 684 685 686 687 688 689 690 691 692 693 694 695 696 697 698 699 700 701 702 703 704 705 706 707 708 709 710 711 712 713 714 715 716 717 718 719 720 721 722 723 724 725 726 727 728 729 730 731 732 733 734 735 736 737 738 739 740 741 742 743 744 745 746 747 748 749 750 751 752 753 754 755 756 757 758 759 760 761 762 763 764 765 766 767 768 769 770 771 772 773 774 775 776 777 778 779 780 781 782 783 784 785 786 787 788 789 790 791 792 793 794 795 796 797 798 799 800 801 802 803 804 805 806 807 808 809 810 811 812 813 814 815 816 817 818 819 820 821 822 823 824 825 826 827 828 829 830 831 832 833 834 835 836 837 838 839 840 841 842 843 844 845 846 847 848 849 850 851 852 853 854 855 856 857 858 859 860 861 862 863 864 865 866 867 868 869 870 871 872 873 874 875 876 877 878 879 880 881 882 883 884 885 886 887 888 889 890 891 892 893 894 895 896 897 898 899 900 901 902 903 904 905 906 907 908 909 910 911 912 913 914 915 916 917 918 919 920 921 922 923 924 925 926 927 928 929 930 931 932 933 934 935 936 937 938 939 940 941 942 943 944 945 946 947 948 949 950 951 952 953 954 955 956 957 958 959 960 961 962 963 964 965 966 967 968 969 970 971 972 973 974 975 976 977 978 979 980 981 982 983 984 985 986 987 988 989 990 991 992 993 994 995 996 997 998 999 1000

Query results SAVE RESULTS EXPLORE DATA ↕


JOB INFORMATION		RESULTS		JSON	EXECUTION DETAILS
Row	avgprice	total	frieght	state	
1	913001.430...	5202955.05...	115342.710...	SP	
2	606849.810...	1824092.66...	86061.41	RJ	
3	330287.000...	683083.760...	45156.2000...	PR	
4	277042.079...	520553.340...	39444.2999...	SC	
5	187296.379...	302603.940...	25772.0299...	DF	
6	531821.450...	1585308.02...	77356.1800...	MG	

Results per page: 50 1 ~ 27 of 27 |< < > >|

PERSONAL HISTORY PROJECT HISTORY REFRESH ^

5.

```
select avg(x.avg_freight_value) as freight_values,
avg(x.diff_estimated_delivery) as estimated_del,
avg(x.time_to_delivery) as time_del,
x.state
from
(SELECT avg(ordt.freight_value) as avg_freight_value,
date_diff(o.order_delivered_customer_date, o.order_purchase_timestamp, DAY) as
time_to_delivery,
date_diff(o.order_delivered_customer_date, o.order_estimated_delivery_date, DAY) as
diff_estimated_delivery,
c.customer_state as state
FROM `target-362415.7008.orders` o
left join `target-362415.7008.order_items` ordt
on o.order_id = ordt.order_id
left join `target-362415.7008.customers` c
on c.customer_id = o.customer_id
group by c.customer_state,time_to_delivery,diff_estimated_delivery
order by state) x
group by x.state
order by time_del desc;
```

 **SANDBOX** Set up billing to upgrade to the full BigQuery experience. [Learn more](#) DISMISS UPGRADE

Explorer + ADD DATA IK Q *Unsaved query + INFO CHAT SHARE

Q Type to search ?

Viewing pinned projects.

- target-362415

RUN SAVE SHARE SCHEDULE MORE Query completed.

```
6 (SELECT avg(ordt.freight_value) as avg_freight_value,
7 date_diff(o.order_delivered_customer_date, o.order_purchase_timestamp, DAY) as time_to_delivery,
8 date_diff(o.order_delivered_customer_date, o.order_estimated_delivery_date, DAY) as diff_estimated_delivery,
9 c.customer_state as state
10 FROM `target-362415.7008.orders` o
11 left join `target-362415.7008.order_items` ordt
12 on o.order_id = ordt.order_id
13 left join `target-362415.7008.customers` c
14 on c.customer_id = o.customer_id
15 group by c.customer_state,time_to_delivery,diff_estimated_delivery
16 order by state) x
17 group by x.state
18 order by time_del desc;
19
```

Press Alt+F1 for Accessibility Options.

Query results SAVE RESULTS EXPLORE DATA ↕

JOB INFORMATION		RESULTS	JSON	EXECUTION DETAILS	
Row	freight_valu...	estimated_d...	time_del	state	
1	42.0111904...	-16.414634...	28.9756097...	RR	
2	34.9273989...	-18.753846...	26.8153846...	AP	
3	33.6205447...	-18.458646...	25.9849624...	AM	
4	36.6435403...	-11.939068...	25.5591397...	PA	
5	22.3095489...	-5.2118055...	25.1898148...	RJ	
6	36.4685418...	-7.8524590...	24.6229508...	AL	

Results per page: 50 1 - 27 of 27 IK < > >|

PERSONAL HISTORY PROJECT HISTORY REFRESH ^

```

select avg(x.diff_estimated_delivery) as Avggestdel,
avg(x.time_to_delivery) as deltime,
x.state
from
(SELECT avg(ordt.freight_value) as avg_freight_value,
date_diff(o.order_delivered_customer_date, o.order_purchase_timestamp, DAY) as
time_to_delivery,
date_diff(o.order_delivered_customer_date, o.order_estimated_delivery_date, DAY) as
diff_estimated_delivery,
c.customer_state as state
FROM `target-362415.7008.orders` o
left join `target-362415.7008.order_items` ordt
on o.order_id = ordt.order_id
left join `target-362415.7008.customers` c
on c.customer_id = o.customer_id
group by c.customer_state,time_to_delivery,diff_estimated_delivery
order by state) x
group by x.state;

```

The screenshot displays a data analytics application interface. On the left, the 'Explorer' panel shows a project named 'target-362415' with a sub-project '7008'. Under '7008', several tables are listed: customers, geolocation, order_items, order_reviews, orders, payments, products, and sellers. The 'payments' table is currently selected.

The main panel shows a SQL query editor with the following code:

```

5 (SELECT avg(ordt.freight_value) as avg_freight_value,
6 date_diff(o.order_delivered_customer_date, o.order_purchase_timestamp, DAY) as time_to_delivery,
7 date_diff(o.order_delivered_customer_date, o.order_estimated_delivery_date, DAY) as diff_estimated_delivery,
8 c.customer_state as state
9 FROM `target-362415.7008.orders` o
10 left join `target-362415.7008.order_items` ordt
11 on o.order_id = ordt.order_id
12 left join `target-362415.7008.customers` c
13 on c.customer_id = o.customer_id
14 group by c.customer_state,time_to_delivery,diff_estimated_delivery
15 order by state) x
16 group by x.state;
17

```

Below the query editor, the 'Query results' section is visible. It includes tabs for 'JOB INFORMATION', 'RESULTS', 'JSON', and 'EXECUTION DETAILS'. The 'RESULTS' tab is active, showing a table with 6 rows and 4 columns: Row, Avggestdel, deltime, and state.

Row	Avggestdel	deltime	state
1	-5.2118055...	25.1898148...	RJ
2	-10.317738...	20.1052631...	RS
3	-6.6431347...	21.0505181...	SP
4	-10.418644...	15.1813559...	DF
5	-10.759433...	16.3325471...	PR
6	-12.323394...	19.5321100...	MT

At the bottom of the interface, there are sections for 'PERSONAL HISTORY' and 'PROJECT HISTORY', along with a 'REFRESH' button. The status bar indicates 'Results per page: 50' and '1 - 27 of 27'.

```

SELECT round(avg(ordt.freight_value),2) as avg_freight_value,
c.customer_state as state
FROM `target-362415.7008.order_items` ordt
left join `target-362415.7008.orders` o
on o.order_id = ordt.order_id
left join `target-362415.7008.customers` c
on c.customer_id = o.customer_id
group by c.customer_state
order by avg_freight_value desc limit 5;

```

The screenshot shows a data analytics application interface. On the left is an 'Explorer' panel with a search bar and a tree view of data sources. The tree view shows a project named 'target-362415' containing an 'External connections' folder and a '7008' folder. The '7008' folder contains several tables: 'customers', 'geolocation', 'order_items', 'order_reviews', 'orders', 'payments' (highlighted), 'products', and 'sellers'. The main area displays a SQL query in a text editor, which is the same query shown in the first block. Below the editor, the 'Query results' section is visible, showing a table with 5 rows of data. The table has columns for 'Row', 'avg_freight_value', and 'state'. The results are ordered by 'avg_freight_value' in descending order. At the bottom of the interface, there are tabs for 'PERSONAL HISTORY' and 'PROJECT HISTORY', and a 'REFRESH' button.

Query results

Row	avg_freight_value	state
1	42.98	RR
2	42.72	PB
3	41.07	RO
4	40.07	AC
5	39.15	PI

```

SELECT round(avg(ordt.freight_value),2) as avg_freight_value,
c.customer_state as state
FROM `target-362415.7008.order_items` ordt
left join `target-362415.7008.orders` o
on o.order_id = ordt.order_id
left join `target-362415.7008.customers` c
on c.customer_id = o.customer_id
group by c.customer_state
order by avg_freight_value asc limit 5;

```

The screenshot shows a data analytics application interface. On the left is an 'Explorer' panel with a search bar and a tree view of pinned projects. The tree shows a project named 'target-362415' containing an 'External connections' folder and a database named '7008'. The database contains several tables: 'customers', 'geolocation', 'order_items', 'order_reviews', 'orders', 'payments' (which is selected), 'products', and 'sellers'. The main area is a query editor with a tab for 'Unsaved query 7'. It contains the same SQL query as shown in the first block. Below the editor, the 'Query results' section is active, displaying a table with 5 rows of data. The table has columns for 'Row', 'avg_freight_value', and 'state'. At the bottom, there are tabs for 'PERSONAL HISTORY' and 'PROJECT HISTORY', and a 'REFRESH' button.

Query results

Row	avg_freight_value	state
1	15.15	SP
2	20.53	PR
3	20.63	MG
4	20.96	RJ
5	21.04	DF

```

SELECT round(avg(date_diff(o.order_delivered_customer_date,
o.order_purchase_timestamp, DAY)),2) as time_to_delivery,
c.customer_state as state
FROM `target-362415.7008.orders` o
left join `target-362415.7008.order_items` ordt
on o.order_id = ordt.order_id
left join `target-362415.7008.customers` c
on c.customer_id = o.customer_id
group by c.customer_state
order by time_to_delivery desc limit 5;

```

The screenshot displays a data analytics application interface. On the left is an 'Explorer' sidebar with a search bar and a tree view of pinned projects. The main area is divided into a query editor at the top and a results section below. The query editor shows a SQL query that calculates the average time to delivery by customer state. The results section, titled 'Query results', contains a table with 5 rows of data. Below the table are tabs for 'PERSONAL HISTORY' and 'PROJECT HISTORY', and a 'REFRESH' button.

Explorer Sidebar:

- target-362415
 - External connections
 - 7008
 - customers
 - geolocation
 - order_items
 - order_reviews
 - orders
 - payments**
 - products
 - sellars

Query Editor:

```

1 SELECT round(avg(date_diff(o.order_delivered_customer_date, o.order_purchase_timestamp, DAY)),2) as time_to_delivery,
2 c.customer_state as state
3 FROM `target-362415.7008.orders` o
4 left join `target-362415.7008.order_items` ordt
5 on o.order_id = ordt.order_id
6 left join `target-362415.7008.customers` c
7 on c.customer_id = o.customer_id
8 group by c.customer_state
9 order by time_to_delivery desc limit 5;
10

```

Query results

Row	time_to_deli...	state
1	27.83	RR
2	27.75	AP
3	25.96	AM
4	23.99	AL
5	23.3	PA

PERSONAL HISTORY PROJECT HISTORY REFRESH

```

SELECT round(avg(date_diff(o.order_delivered_customer_date,
o.order_purchase_timestamp, DAY)),2) as time_to_delivery,
c.customer_state as state
FROM `target-362415.7008.orders` o
left join `target-362415.7008.order_items` ordt
on o.order_id = ordt.order_id
left join `target-362415.7008.customers` c
on c.customer_id = o.customer_id
group by c.customer_state
order by time_to_delivery asc limit 5;

```

The screenshot shows a web-based data analytics tool. On the left is an 'Explorer' sidebar with a search bar and a tree view of pinned projects. The tree shows a project named 'target-362415' containing a folder '7008' with tables: 'customers', 'geolocation', 'order_items', 'order_reviews', 'orders', 'payments' (highlighted), 'products', and 'sellers'. The main area is divided into a query editor at the top and a results section below. The query editor contains the SQL code from the previous block. Below the editor, the 'Query results' section is active, displaying a table with 5 rows of data. At the bottom, there are tabs for 'PERSONAL HISTORY' and 'PROJECT HISTORY', and a 'REFRESH' button.

Query results

Row	time_to_del...	state
1	8.26	SP
2	11.48	PR
3	11.52	MG
4	12.5	DF
5	14.52	SC

```

select round(avg(date_diff(o.order_estimated_delivery_date,
o.order_delivered_customer_date, DAY)),2) as diff_estimated_delivery,
c.customer_state as state
FROM `target-362415.7008.orders` o
left join `target-362415.7008.order_items` ordt
on o.order_id = ordt.order_id
left join `target-362415.7008.customers` c
on c.customer_id = o.customer_id
group by c.customer_state
order by diff_estimated_delivery desc limit 5;

```

The screenshot displays a data analytics application interface. On the left is an 'Explorer' sidebar with a search bar and a tree view of pinned projects. The 'target-362415' project is expanded, showing sub-projects like '7008' which contains tables: 'customers', 'geolocation', 'order_items', 'order_reviews', 'orders', 'payments' (highlighted), 'products', and 'sellers'. The main area features a SQL query editor with a toolbar (RUN, SAVE, SHARE, SCHEDULE, MORE) and a query window containing the same SQL code as the first block. Below the editor, the 'Query results' section is active, showing a table with 5 rows of data. At the bottom, there are tabs for 'PERSONAL HISTORY' and 'PROJECT HISTORY', and a 'REFRESH' button.

Query results

Row	diff_estimat...	state
1	20.01	AC
2	19.08	RO
3	18.98	AM
4	17.44	AP
5	17.43	RR


```

select round(avg(date_diff(o.order_estimated_delivery_date,
o.order_delivered_customer_date, DAY)),2) as diff_estimated_delivery,
c.customer_state as state
FROM `target-362415.7008.orders` o
left join `target-362415.7008.order_items` ordt
on o.order_id = ordt.order_id
left join `target-362415.7008.customers` c
on c.customer_id = o.customer_id
group by c.customer_state
order by diff_estimated_delivery asc limit 5;

```

The screenshot displays a data analytics application interface. On the left is an 'Explorer' panel with a search bar and a tree view of pinned projects. The main area is divided into a query editor at the top and a 'Query results' section at the bottom. The query editor contains a SQL query that calculates the average difference in days between the estimated and delivered delivery dates, grouped by customer state, and orders the results by this difference in ascending order, limiting the output to 5 rows. The 'Query results' section shows a table with 5 rows of data, including columns for row number, the calculated difference, and the customer state. The interface also includes various toolbars for query execution, saving, and sharing, as well as tabs for different queries.

Query results

Row	diff_estimat...	state
1	7.98	AL
2	9.11	MA
3	9.17	SE
4	9.77	ES
5	10.12	BA

6.

```
SELECT count(ord.order_id) as ordercount,
extract(month from ord.order_purchase_timestamp) as month,
extract(year from ord.order_purchase_timestamp) as year,
p.payment_type as payment
from `target-362415.7008.orders` ord
left join `target-362415.7008.payments` p
on ord.order_id = p.order_id
group by p.payment_type,year,month;
```

The screenshot displays a data analytics application interface. On the left is an 'Explorer' sidebar with a search bar and a tree view of pinned projects. The tree shows a project named 'target-362415' containing a dataset '7008', which in turn contains several tables: 'customers', 'geolocation', 'order_items', 'order_reviews', 'orders', 'payments' (highlighted), 'products', and 'sellers'. The main area is divided into a top query editor and a bottom results section. The query editor shows a SQL query (lines 1-9) that counts orders by payment type, year, and month. The results section, titled 'Query results', shows a table with 6 rows of data. The table has columns for 'Row', 'ordercount', 'month', 'year', and 'payment'. The data shows various payment types like 'UPI', 'credit_card', and 'voucher' across different months and years. At the bottom, there are tabs for 'PERSONAL HISTORY' and 'PROJECT HISTORY', and a 'REFRESH' button.

Query results

Row	ordercount	month	year	payment
1	1509	11	2017	UPI
2	4377	12	2017	credit_card
3	1325	2	2018	UPI
4	5897	11	2017	credit_card
5	202	4	2017	voucher
6	3086	7	2017	credit_card

```
SELECT count(p.order_id) as ordercount,  
p.payment_type  
FROM `target-362415.7008.payments` p  
group by p.payment_type  
order by ordercount desc;
```

The screenshot displays a data analytics application interface. On the left is an 'Explorer' sidebar with a search bar and a list of pinned projects, including 'target-362415'. The main workspace is titled 'Unsaved query' and contains a SQL query. Below the query editor, the 'Query results' section is active, showing a table with 5 rows of data. The table has columns for 'Row', 'ordercount', and 'payment_type'. At the bottom of the interface, there are tabs for 'PERSONAL HISTORY' and 'PROJECT HISTORY', along with a 'REFRESH' button.

SQL Query:

```
1 SELECT count(p.order_id) as ordercount,  
2 p.payment_type  
3 FROM `target-362415.7008.payments` p  
4 group by p.payment_type  
5 order by ordercount desc;
```

Query results

Row	ordercount	payment_type
1	76795	credit_card
2	19784	UPI
3	5775	voucher
4	1529	debit_card
5	3	not_defined

Initial Analysis/Actionable insights:

1. The order count has significantly increased over the years.
2. The order count peaked in November in 2017 but we are not seeing any seasonality trends.
3. We can observe that the shopping numbers are high during the afternoon.
4. Observed that in SP state the order count decreased to half in the month of October.
5. DF and ES state have the least number of orders.
6. SP and MP have the highest number of orders.
7. SP has the highest number of customers and RR has the least number of customers.
8. Cost percentage increased nearly 27 percent in 2018 then 2017 from the month January to August
9. The average price and freight value is high for state SP and low for state RR.
10. Estimated delivery is negative means the delivery is getting done before the estimated time.
11. The average delivery time is very high in state RR compared to the state DF which has least delivery time.
12. RR,PB,RO,AC and PI have the highest freight value.
13. SP,PR,MG,RJ and DF have the least freight value.
14. RR,AP,AM,AL and PA have the highest delivery time.
15. SP,PR,MG,DF and SC have the least delivery time.
16. AC,RO,AM,AP and RR have the highest estimated time difference. Means these states are delivering orders very earlier then the estimated delivery time.
17. AL,MA,AC,ES and BA have the least estimated time difference. Means these states are delivering orders when the estimated delivery time is nearer.
18. Customers are more likely to use credit cards for payments.
19. Customers are less likely to use debit cards for payments.

Recommendations:

1. The shopping numbers are higher in the afternoon so we can keep more sales persons in that time period.
2. We can give more offers from DF and ES to increase the order count.
3. We can increase the price in SP and MP as these 2 states have the highest number of orders.
4. We can initiate some offers in state RR for attracting more customers as it has the least number of customers.
5. The cost percentage has increased 27 percent from 2017 to 2018 which is on a higher side. We can offer some discounts to loyal customers.
6. In state RR the average delivery time is very high which might be the reason for less no of orders from this place. We can reduce the delivery time for the orders.
7. Similar to state RR other states are there who have high delivery time which can be reduced.

8. States having the highest estimated time difference means we can mention reduced estimated delivery time which will help our customers a relief by seeing nearer estimated delivery date.
9. To promote other payment methods we can roll out small offers for payment methods which are least used. Like debit card, vouchers.
10. As we can see, customers are using credit cards and UPI more for payments so we should deploy card payment machines and UPI payment methods for easy transactions.
11. As customers are using credit cards there may be more chances of EMIs so we can partner with banks for no cost EMI or less interest rates options.