

# FACTS: TABLE SUMMARIZATION VIA OFFLINE TEMPLATE GENERATION WITH AGENTIC WORKFLOWS

000  
001  
002  
003  
004  
005 **Anonymous authors**  
006 Paper under double-blind review  
007  
008  
009  
010  
011  
012  
013  
014  
015  
016  
017  
018  
019  
020  
021  
022  
023  
024  
025  
026  
027  
028  
029  
030  
031  
032  
033  
034  
035  
036  
037  
038  
039  
040  
041  
042  
043  
044  
045  
046  
047  
048  
049  
050  
051  
052  
053  
054  
055  
056  
057  
058  
059  
060  
061  
062  
063  
064  
065  
066  
067  
068  
069  
070  
071  
072  
073  
074  
075  
076  
077  
078  
079  
080  
081  
082  
083  
084  
085  
086  
087  
088  
089  
090  
091  
092  
093  
094  
095  
096  
097  
098  
099  
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  
559  
560  
561  
562  
563  
564  
565  
566  
567  
568  
569  
569  
570  
571  
572  
573  
574  
575  
576  
577  
578  
579  
579  
580  
581  
582  
583  
584  
585  
586  
587  
588  
589  
589  
590  
591  
592  
593  
594  
595  
596  
597  
598  
599  
599  
600  
601  
602  
603  
604  
605  
606  
607  
608  
609  
609  
610  
611  
612  
613  
614  
615  
616  
617  
618  
619  
619  
620  
621  
622  
623  
624  
625  
626  
627  
628  
629  
629  
630  
631  
632  
633  
634  
635  
636  
637  
638  
639  
639  
640  
641  
642  
643  
644  
645  
646  
647  
648  
649  
649  
650  
651  
652  
653  
654  
655  
656  
657  
658  
659  
659  
660  
661  
662  
663  
664  
665  
666  
667  
668  
669  
669  
670  
671  
672  
673  
674  
675  
676  
677  
678  
679  
679  
680  
681  
682  
683  
684  
685  
686  
687  
688  
689  
689  
690  
691  
692  
693  
694  
695  
696  
697  
698  
699  
699  
700  
701  
702  
703  
704  
705  
706  
707  
708  
709  
709  
710  
711  
712  
713  
714  
715  
716  
717  
718  
719  
719  
720  
721  
722  
723  
724  
725  
726  
727  
728  
729  
729  
730  
731  
732  
733  
734  
735  
736  
737  
738  
739  
739  
740  
741  
742  
743  
744  
745  
746  
747  
748  
749  
749  
750  
751  
752  
753  
754  
755  
756  
757  
758  
759  
759  
760  
761  
762  
763  
764  
765  
766  
767  
768  
769  
769  
770  
771  
772  
773  
774  
775  
776  
777  
778  
779  
779  
780  
781  
782  
783  
784  
785  
786  
787  
788  
789  
789  
790  
791  
792  
793  
794  
795  
796  
797  
798  
799  
799  
800  
801  
802  
803  
804  
805  
806  
807  
808  
809  
809  
810  
811  
812  
813  
814  
815  
816  
817  
818  
819  
819  
820  
821  
822  
823  
824  
825  
826  
827  
828  
829  
829  
830  
831  
832  
833  
834  
835  
836  
837  
838  
839  
839  
840  
841  
842  
843  
844  
845  
846  
847  
848  
849  
849  
850  
851  
852  
853  
854  
855  
856  
857  
858  
859  
859  
860  
861  
862  
863  
864  
865  
866  
867  
868  
869  
869  
870  
871  
872  
873  
874  
875  
876  
877  
878  
879  
879  
880  
881  
882  
883  
884  
885  
886  
887  
888  
889  
889  
890  
891  
892  
893  
894  
895  
896  
897  
898  
899  
899  
900  
901  
902  
903  
904  
905  
906  
907  
908  
909  
909  
910  
911  
912  
913  
914  
915  
916  
917  
918  
919  
919  
920  
921  
922  
923  
924  
925  
926  
927  
928  
929  
929  
930  
931  
932  
933  
934  
935  
936  
937  
938  
939  
939  
940  
941  
942  
943  
944  
945  
946  
947  
948  
949  
949  
950  
951  
952  
953  
954  
955  
956  
957  
958  
959  
959  
960  
961  
962  
963  
964  
965  
966  
967  
968  
969  
969  
970  
971  
972  
973  
974  
975  
976  
977  
978  
979  
979  
980  
981  
982  
983  
984  
985  
986  
987  
988  
989  
989  
990  
991  
992  
993  
994  
995  
996  
997  
998  
999  
1000  
1001  
1002  
1003  
1004  
1005  
1006  
1007  
1008  
1009  
1009  
1010  
1011  
1012  
1013  
1014  
1015  
1016  
1017  
1018  
1019  
1019  
1020  
1021  
1022  
1023  
1024  
1025  
1026  
1027  
1028  
1029  
1029  
1030  
1031  
1032  
1033  
1034  
1035  
1036  
1037  
1038  
1039  
1039  
1040  
1041  
1042  
1043  
1044  
1045  
1046  
1047  
1048  
1049  
1049  
1050  
1051  
1052  
1053  
1054  
1055  
1056  
1057  
1058  
1059  
1059  
1060  
1061  
1062  
1063  
1064  
1065  
1066  
1067  
1068  
1069  
1069  
1070  
1071  
1072  
1073  
1074  
1075  
1076  
1077  
1078  
1079  
1079  
1080  
1081  
1082  
1083  
1084  
1085  
1086  
1087  
1088  
1089  
1089  
1090  
1091  
1092  
1093  
1094  
1095  
1096  
1097  
1098  
1098  
1099  
1099  
1100  
1101  
1102  
1103  
1104  
1105  
1106  
1107  
1108  
1109  
1109  
1110  
1111  
1112  
1113  
1114  
1115  
1116  
1117  
1118  
1119  
1119  
1120  
1121  
1122  
1123  
1124  
1125  
1126  
1127  
1128  
1129  
1129  
1130  
1131  
1132  
1133  
1134  
1135  
1136  
1137  
1138  
1139  
1139  
1140  
1141  
1142  
1143  
1144  
1145  
1146  
1147  
1148  
1149  
1149  
1150  
1151  
1152  
1153  
1154  
1155  
1156  
1157  
1158  
1159  
1159  
1160  
1161  
1162  
1163  
1164  
1165  
1166  
1167  
1168  
1169  
1169  
1170  
1171  
1172  
1173  
1174  
1175  
1176  
1177  
1178  
1179  
1179  
1180  
1181  
1182  
1183  
1184  
1185  
1186  
1187  
1188  
1189  
1189  
1190  
1191  
1192  
1193  
1194  
1195  
1196  
1197  
1198  
1198  
1199  
1199  
1200  
1201  
1202  
1203  
1204  
1205  
1206  
1207  
1208  
1209  
1209  
1210  
1211  
1212  
1213  
1214  
1215  
1216  
1217  
1218  
1219  
1219  
1220  
1221  
1222  
1223  
1224  
1225  
1226  
1227  
1228  
1229  
1229  
1230  
1231  
1232  
1233  
1234  
1235  
1236  
1237  
1238  
1239  
1239  
1240  
1241  
1242  
1243  
1244  
1245  
1246  
1247  
1248  
1249  
1249  
1250  
1251  
1252  
1253  
1254  
1255  
1256  
1257  
1258  
1259  
1259  
1260  
1261  
1262  
1263  
1264  
1265  
1266  
1267  
1268  
1269  
1269  
1270  
1271  
1272  
1273  
1274  
1275  
1276  
1277  
1278  
1279  
1279  
1280  
1281  
1282  
1283  
1284  
1285  
1286  
1287  
1288  
1289  
1289  
1290  
1291  
1292  
1293  
1294  
1295  
1296  
1297  
1298  
1298  
1299  
1299  
1300  
1301  
1302  
1303  
1304  
1305  
1306  
1307  
1308  
1309  
1309  
1310  
1311  
1312  
1313  
1314  
1315  
1316  
1317  
1318  
1319  
1319  
1320  
1321  
1322  
1323  
1324  
1325  
1326  
1327  
1328  
1329  
1329  
1330  
1331  
1332  
1333  
1334  
1335  
1336  
1337  
1338  
1339  
1339  
1340  
1341  
1342  
1343  
1344  
1345  
1346  
1347  
1348  
1349  
1349  
1350  
1351  
1352  
1353  
1354  
1355  
1356  
1357  
1358  
1359  
1359  
1360  
1361  
1362  
1363  
1364  
1365  
1366  
1367  
1368  
1369  
1369  
1370  
1371  
1372  
1373  
1374  
1375  
1376  
1377  
1378  
1379  
1379  
1380  
1381  
1382  
1383  
1384  
1385  
1386  
1387  
1388  
1389  
1389  
1390  
1391  
1392  
1393  
1394  
1395  
1396  
1397  
1398  
1398  
1399  
1399  
1400  
1401  
1402  
1403  
1404  
1405  
1406  
1407  
1408  
1409  
1409  
1410  
1411  
1412  
1413  
1414  
1415  
1416  
1417  
1418  
1419  
1419  
1420  
1421  
1422  
1423  
1424  
1425  
1426  
1427  
1428  
1429  
1429  
1430  
1431  
1432  
1433  
1434  
1435  
1436  
1437  
1438  
1439  
1439  
1440  
1441  
1442  
1443  
1444  
1445  
1446  
1447  
1448  
1449  
1449  
1450  
1451  
1452  
1453  
1454  
1455  
1456  
1457  
1458  
1459  
1459  
1460  
1461  
1462  
1463  
1464  
1465  
1466  
1467  
1468  
1469  
1469  
1470  
1471  
1472  
1473  
1474  
1475  
1476  
1477  
1478  
1479  
1479  
1480  
1481  
1482  
1483  
1484  
1485  
1486  
1487  
1488  
1489  
1489  
1490  
1491  
1492  
1493  
1494  
1495  
1496  
1497  
1498  
1498  
1499  
1499  
1500  
1501  
1502  
1503  
1504  
1505  
1506  
1507  
1508  
1509  
1509  
1510  
1511  
1512  
1513  
1514  
1515  
1516  
1517  
1518  
1519  
1519  
1520  
1521  
1522  
1523  
1524  
1525  
1526  
1527  
1528  
1529  
1529  
1530  
1531  
1532  
1533  
1534  
1535  
1536  
1537  
1538  
1539  
1539  
1540  
1541  
1542  
1543  
1544  
1545  
1546  
1547  
1548  
1549  
1549  
1550  
1551  
1552  
1553  
1554  
1555  
1556  
1557  
1558  
1559  
1559  
1560  
1561  
1562  
1563  
1564  
1565  
1566  
1567  
1568  
1569  
1569  
1570  
1571  
1572  
1573  
1574  
1575  
1576  
1577  
1578  
1579  
1579  
1580  
1581  
1582  
1583  
1584  
1585  
1586  
1587  
1588  
1589  
1589  
1590  
1591  
1592  
1593  
1594  
1595  
1596  
1597  
1598  
1598  
1599  
1599  
1600  
1601  
1602  
1603  
1604  
1605  
1606  
1607  
1608  
1609  
1609  
1610  
1611  
1612  
1613  
1614  
1615  
1616  
1617  
1618  
1619  
1619  
1620  
1621  
1622  
1623  
1624  
1625  
1626  
1627  
1628  
1629  
1629  
1630  
1631  
1632  
1633  
1634  
1635  
1636  
1637  
1638  
1639  
1639  
1640  
1641  
1642  
1643  
1644  
1645  
1646  
1647  
1648  
1649  
1649  
1650  
1651  
1652  
1653  
1654  
1655  
1656  
1657  
1658  
1659  
1659  
1660  
1661  
1662  
1663  
1664  
1665  
1666  
1667  
1668  
1669  
1669  
1670  
1671  
1672  
1673  
1674  
1675  
1676  
1677  
1678  
1679  
1679  
1680  
1681  
1682  
1683  
1684  
1685  
1686  
1687  
1688  
1689  
1689  
1690  
1691  
1692  
1693  
1694  
1695  
1696  
1697  
1698  
1698  
1699  
1699  
1700  
1701  
1702  
1703  
1704  
1705  
1706  
1707  
1708  
1709  
1709  
1710  
1711  
1712  
1713  
1714  
1715  
1716  
1717  
1718  
1719  
1719  
1720  
1721  
1722  
1723  
1724  
1725  
1726  
1727  
1728  
1729  
1729  
1730  
1731  
1732  
1733  
1734  
1735  
1736  
1737  
1738  
1739  
1739  
1740  
1741  
1742  
1743  
1744  
1745  
1746  
1747  
1748  
1749  
1749  
1750  
1751  
1752  
1753  
1754  
1755  
1756  
1757  
1758  
1759  
1759  
1760  
1761  
1762  
1763  
1764  
1765  
1766  
1767  
1768  
1769  
1769  
1770  
1771  
1772  
1773  
1774  
1775  
1776  
1777  
1778  
1779  
1779  
1780  
1781  
1782  
1783  
1784  
1785  
1786  
1787  
1788  
1789  
1789  
1790  
1791  
1792  
1793  
1794  
1795  
1796  
1797  
1798  
1798  
1799  
1799  
1800  
1801  
1802  
1803  
1804  
1805  
1806  
1807  
1808  
1809  
1809  
1810  
1811  
1812  
1813  
1814  
1815  
1816  
1817  
1818  
1819  
1819  
1820  
1821  
1822  
1823  
1824  
1825  
1826  
1827  
1828  
1829  
1829  
1830  
1831  
1832  
1833  
1834  
1835  
1836  
1837  
1838  
1839  
1839  
1840  
1841  
1842  
1843  
1844  
1845  
1846  
1847  
1848  
1849  
1849  
1850  
1851  
1852  
1853  
1854  
1855  
1856  
1857  
1858  
1859  
1859  
1860  
1861  
1862  
1863  
1864  
1865  
1866  
1867  
1868  
1869  
1869  
1870  
1871  
1872  
1873  
1874  
1875  
1876  
1877  
1878  
1879  
1879  
1880  
1881  
1882  
1883  
1884  
1885  
1886  
1887  
1888  
1889  
1889  
1890  
1891  
1892  
1893  
1894  
1895  
1896  
1897  
1898  
1898  
1899  
1899  
1900  
1901  
1902  
1903  
1904  
1905<br

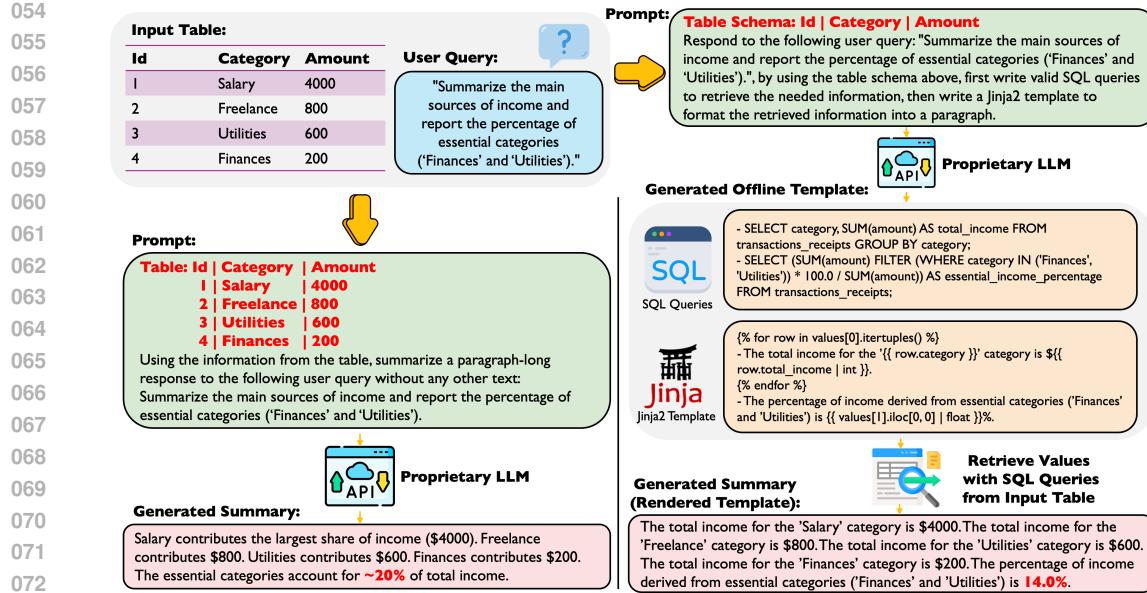


Figure 1: Comparison between DirectSumm (Zhang et al., 2024) (left) and our proposed FACTS framework (right). DirectSumm prompts a large language model (LLM) with the full table and query, which may produce hallucinated values, exposes all table records to external services, and requires regeneration for each new table even under the same schema and query. In contrast, FACTS generates a reusable offline template consisting of schema-aware SQL queries and a Jinja2 template. The SQL queries retrieve precise values through execution, while the Jinja2 template renders natural language summaries, ensuring accuracy, reusability, scalability, and privacy compliance.

for ten yearly tables, with all values revealed to the model, leading to inefficiency and privacy risks, as illustrated in Figure 1 (left).

To address these challenges, we introduce **FACTS**, a *Fast, Accurate, and Privacy-Compliant Table Summarization approach via Offline Template Generation*. FACTS employs an agentic workflow with three stages. First, it generates schema-aware guided questions and filtering rules to clarify user query intent. Second, it synthesizes SQL queries to extract relevant information from tables. Third, it produces a Jinja2 template to render SQL outputs into natural language. Crucially, FACTS integrates an LLM Council, an ensemble of LLMs iteratively validating and refining outputs at each stage. This feedback loop ensures correctness, consistency, and usability of the generated artifacts. The final product, an offline template composed of SQL queries and a Jinja2 template, can be reused across any tables with the same schema for a given query. Returning to our example, an offline template produced by FACTS can summarize gross income across ten yearly tables, avoiding repeated LLM calls while ensuring accurate and privacy-compliant outputs (Figure 1 (right)). **To the best of our knowledge, FACTS introduces the first agentic framework that automates offline template generation for query-focused table summarization.**

We evaluate FACTS on three public benchmarks: FeTaQA (Nan et al., 2022), QTSumm (Zhao et al., 2023), and QFMTS (Zhang et al., 2024). Experimental results show that FACTS consistently outperforms representative baselines, demonstrating its practicality for real-world query-focused table summarization.

In summary, our contributions are as follows:

- We propose offline template generation, which produces reusable and schema-specific templates in a privacy-compliant manner, enabling scalability to large tables and efficiency across recurring queries.
- We design FACTS, an agentic workflow that integrates guided question generation, SQL synthesis, and Jinja2 rendering, supported by iterative feedback loops to ensure correctness.
- We demonstrate the practicality of FACTS through comprehensive experiments on FeTaQA, QTSumm, and QFMTS, showing promising improvements over representative baselines.

108  
 109 Table 1: Comparison of paradigms for query-focused table summarization. Only FACTS satisfies all  
 110 four desired properties. TaPERA and SPaGe produce *partially reusable* plans, denoted as  $\sim$ .

Method	Reusable	Scalable	Accurate	Privacy-Compliant
Table-to-Text Models	$\times$	$\times$	$\times$	$\times$
Prompt-Based Models	$\times$	$\times$	$\times$	$\times$
Binder (Cheng et al., 2023)	$\times$	$\times$	$\checkmark$	$\times$
Dater (Ye et al., 2023)	$\times$	$\times$	$\checkmark$	$\times$
TaPERA (Zhao et al., 2024)	$\sim$	$\times$	$\checkmark$	$\times$
SPaGe (Zhang et al., 2025)	$\sim$	$\times$	$\checkmark$	$\times$
FACTS (ours)	$\checkmark$	$\checkmark$	$\checkmark$	$\checkmark$

## 2 RELATED WORK

This section reviews prior work related to our study. We first situate query-focused table summarization within the broader landscape of table summarization and question answering. We then survey existing approaches and compare these paradigms against our proposed framework.

**Query-Focused Table Summarization.** Research on table-to-text generation has primarily aimed at transforming structured tables into natural language statements or summaries (Parikh et al., 2020; Chen et al., 2020; Cheng et al., 2022b; Lebret et al., 2016; Moosavi et al., 2021; Suadaa et al., 2021). These works typically target either single-sentence descriptions or domain-specific summaries, with the main goal of improving fluency and factual consistency. However, such outputs are not tailored to a user’s specific information needs. In contrast, table question answering (Pasupat & Liang, 2015; Iyyer et al., 2017; Nan et al., 2022) has focused on answering precise fact-based queries, usually returning short values or entities. While table question answering captures query intent, it lacks the ability to provide longer-form reasoning or explanatory summaries. To address this gap, Zhao et al. (2023) introduced the task of query-focused table summarization, where a model generates a narrative-style summary conditioned on both the table and a user query. Compared to generic table summarization, query-focused table summarization explicitly accounts for diverse user intents, and compared to table question answering, it produces extended summaries rather than minimal answers.

**Existing Approaches.** Existing work can be broadly grouped into three categories. **(1) Table-to-text models** adapt language models to better capture table structure and reasoning. TAPEX (Liu et al., 2022b) extends BART with large-scale synthetic SQL execution data, improving compositional reasoning. ReasTAP (Zhao et al., 2022) follows a similar idea but uses synthetic QA corpora to enhance logical understanding. OmniTab (Jiang et al., 2022) combines both natural and synthetic QA signals for more robust pretraining. FORTAP (Cheng et al., 2022a) leverages spreadsheet formulas as supervision to strengthen numerical reasoning. PLOG (Liu et al., 2022a) introduces a two-stage strategy: first generating logical forms from tables, then converting them into natural language, to improve logical faithfulness in summaries. **(2) Prompt-based models** instead rely directly on large language models (LLMs) with carefully designed prompting. ReFactor (Zhao et al., 2023) extracts query-relevant facts and concatenates them with the query to guide generation. DirectSumm (Zhang et al., 2024) produces summaries in a single step, synthesizing text directly from the table and query. Reason-then-Summ (Zhang et al., 2024) decomposes the task into two stages, first retrieving relevant facts and then composing longer summaries. **(3) Agentic frameworks** use external tools such as SQL or Python to ensure accuracy. Binder (Cheng et al., 2023) translates the input query into executable programs, often SQL, to ground results in computation. Dater (Ye et al., 2023) decomposes complex queries into smaller sub-queries, executes them individually, and aggregates their outputs. TaPERA (Zhao et al., 2024) builds natural language plans that are converted into Python programs for execution before aggregation. SPaGe (Zhang et al., 2025) moves beyond free-form plans by introducing structured representations and graph-based execution, improving reliability in multi-table scenarios. Table 1 contrasts our proposed FACTS with representative methods using four criteria. *Reusable*: artifacts applicable to new tables with the same schema; *Scalable*: ability to handle very large tables without feeding all rows; *Accurate*: correctness via executable programs; *Privacy-Compliant*: avoiding exposure of raw table content to LLMs. Most prior methods fall short on one or more dimensions: table-to-text and prompt-based models lack all four; agentic frameworks

162 Example 1: An offline template generated by FACTS on the QFMTS dataset (Zhang et al., 2024).  
 163 The SQL query retrieves the top three accounts by savings balance, and the Jinja2 template renders  
 164 the results into natural language.

```

165 SQL Queries:
166 - SELECT a."name", s."balance"
167   FROM "ACCOUNTS" a
168   JOIN "SAVINGS" s
169     ON CAST(a."custid" AS DOUBLE) = s."custid"
170   ORDER BY s."balance" DESC, a."name" ASC
171   LIMIT 3;
172 Jinja2 Template:
173   {% if values and values|length > 0 %}
174     The three accounts with the highest savings balances are:
175     {% for row in values %}
176       - {{ row["name"] }} with a savings balance of {{ row["balance"] }}
177     {% endfor %}
178     Overall, these represent the top savers by balance in the dataset.
179     {% else %}
180       No results were found for the requested top savings accounts.
181     {% endif %}
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

```

improve accuracy but sacrifice scalability and privacy; and plan-based methods, such as TaPERA and SPaGe, yield only partially reusable plans. FACTS is the only approach satisfying all four desired properties.

### 3 METHODOLOGY

To avoid ambiguity, we first clarify the terminology used in this section. A `user query` denotes the natural language input provided by the user, which specifies an information need over one or more tables and may include rich contextual details. An `SQL query` refers to executable code generated by our method to retrieve the information required to satisfy the user query. A `Jinja2 template` is a rendering program that verbalizes SQL outputs into natural language. An `offline template` is the composite artifact introduced in this work, bundling one or more SQL queries together with a `Jinja2 template`. Unless otherwise specified, the term `schema` refers to the structural metadata of the table, e.g., column names and data types, rather than raw values. Finally, a `summary` denotes the final natural language output returned to the user after executing the SQL queries and rendering the `Jinja2 template`. The remainder of this section is structured as follows: Section 3.1 introduces the concept of offline templates and motivates their reusability; Section 3.2 details the LLM Council, which provides iterative validation and feedback; and Section 3.3 presents the complete FACTS framework and its three interconnected modules.

#### 3.1 OFFLINE TEMPLATE

Formally, an offline template is defined as a composite artifact consisting of (1) one or more schema-aware SQL queries that retrieve relevant facts from the underlying tables, and (2) a `Jinja2 template` that transforms the retrieved outputs into a natural language summary. Crucially, offline templates are bound to both the table schema and the user query semantics. Once generated, the same offline template can be directly applied to any table sharing the same schema and answering the same user query or semantically similar queries, enabling reusability across tables that differ only in values, e.g., multiple years of financial records or multiple patients' health records. **In this work, we define template reusability under an identical schema, without considering schema drift or renamed columns.** This design avoids repeated LLM inference, provides efficiency through lightweight SQL execution, and ensures privacy compliance by never exposing raw table values to LLMs. Example 1 illustrates a real template generated by FACTS on the QFMTS dataset (Zhang et al., 2024). Here, the SQL query selects the top three accounts by savings balance from the `ACCOUNTS` and `SAVINGS` tables, and the `Jinja2 template` verbalizes the results into a coherent narrative. This example demonstrates offline

---

216 **Algorithm 1** LLM Council: Evaluate-and-Refine

---

217 **Input:** artifact  $A$  (e.g., a guided question, SQL query, Jinja2 template, or summary), context  $\mathcal{X}$   
 218 (schema, guidance, execution logs), model set  $\mathcal{C} = \{M_1, \dots, M_m\}$   
 219 **Output:** decision  $\text{DEC} \in \{\text{YES}, \text{NO}\}$ , consensus feedback  $\text{FB}$

```

220 1:  $\mathcal{R} \leftarrow \emptyset$                                  $\triangleright$  per-model results
221 2: for  $M \in \mathcal{C}$  do                          $\triangleright$  independent judgments
222   3:    $p \leftarrow \text{BUILD_PROMPT}(A, \mathcal{X})$ 
223   4:    $o \leftarrow \text{LLM_CALL}(M, p)$                    $\triangleright$  LLM call
224   5:    $(d, f) \leftarrow \text{PARSE}(o)$                    $\triangleright d \in \{\text{YES}, \text{NO}\}, f = \text{brief feedback}$ 
225   6:    $\mathcal{R} \leftarrow \mathcal{R} \cup \{(d, f)\}$ 
226 7:  $\text{DEC} \leftarrow \text{MAJORITY_VOTE}(\{d : (d, f) \in \mathcal{R}\})$ 
227 8:  $\text{FB} \leftarrow \text{AGGREGATE}(\{f : (d, f) \in \mathcal{R}\}, \text{DEC})$            $\triangleright$  short consensus rationale
228 9: return  $(\text{DEC}, \text{FB})$ 

```

---

231 templates are executable and reusable artifacts that faithfully capture user intent and generalize across  
 232 tables with the same schema and query semantics.

233 

### 3.2 LLM COUNCIL

236 The LLM Council is an ensemble of LLMs that collaboratively validate intermediate outputs at each  
 237 stage of the FACTS framework. Rather than relying on a single model, the Council prompts multiple  
 238 heterogeneous LLMs, each of which independently produces a structured judgment (YES/NO) and  
 239 brief feedback. A majority-voting scheme determines whether a candidate artifact is accepted, while  
 240 the collected feedback is aggregated into a consensus explanation that guides iterative refinement.  
 241 The Council provides feedback in four places: (1) evaluating guided questions and filtering rules, (2)  
 242 validating generated SQL queries, (3) checking alignment between SQL results and Jinja2 templates,  
 243 and (4) assessing whether the final summary satisfies the user query. These validation steps will be  
 244 described in detail in the next subsection. This mechanism reduces reliance on any single model,  
 245 mitigates hallucinations, and ensures correctness and usability of generated artifacts.

246 Algorithm 1 presents the Council’s evaluate-and-refine procedure in pseudocode. For each candidate  
 247 artifact, a task-specific prompt is built from the artifact and its context, e.g., table schema, guided  
 248 questions, execution logs, and passed independently to every model in the ensemble. Their responses  
 249 are parsed into decisions and feedback, after which majority voting determines the overall acceptance  
 250 decision, and aggregated feedback provides a concise rationale to guide refinement. Full prompt  
 251 templates are included in Appendix A.1.

252 

### 3.3 FACTS FRAMEWORK

254 The FACTS framework is composed of three interconnected stages, shown in Figure 2, with full  
 255 pseudocode provided in Appendix A.2. At each stage, outputs generated by the LLM agent are  
 256 validated by the LLM Council introduced in Section 3.2, which provides structured feedback and  
 257 guides iterative refinement.

258 **Stage 1: Schema-Guided Specification and Filtering.** Given the user query and table schema, the  
 259 agent first generates schema-aware clarifications in two complementary forms: (i) guided questions  
 260 that identify which columns, relationships, and operations are relevant, and (ii) filtering rules that  
 261 specify which rows or categorical values should be excluded. Crucially, the LLM never accesses the  
 262 raw table contents. Instead, it proposes filtering rules in abstract form, e.g., “exclude rows where  
 263 `category='expense'`”, which are later expressed as WHERE clauses in SQL. This ensures the  
 264 filtering process remains privacy-compliant and syntactically verifiable. For example, in the financial  
 265 scenario introduced earlier, the agent may generate rules that remove irrelevant transaction categories,  
 266 e.g., “exclude expense transactions”, before producing summaries of gross income. The resulting  
 267 schema-guided specifications serve as input to SQL synthesis in the next stage.

268 **Stage 2: SQL Queries Generation.** Using the approved specifications from Stage 1, the agent  
 269 synthesizes one or more candidate SQL queries. These SQL queries integrate the filtering rules as

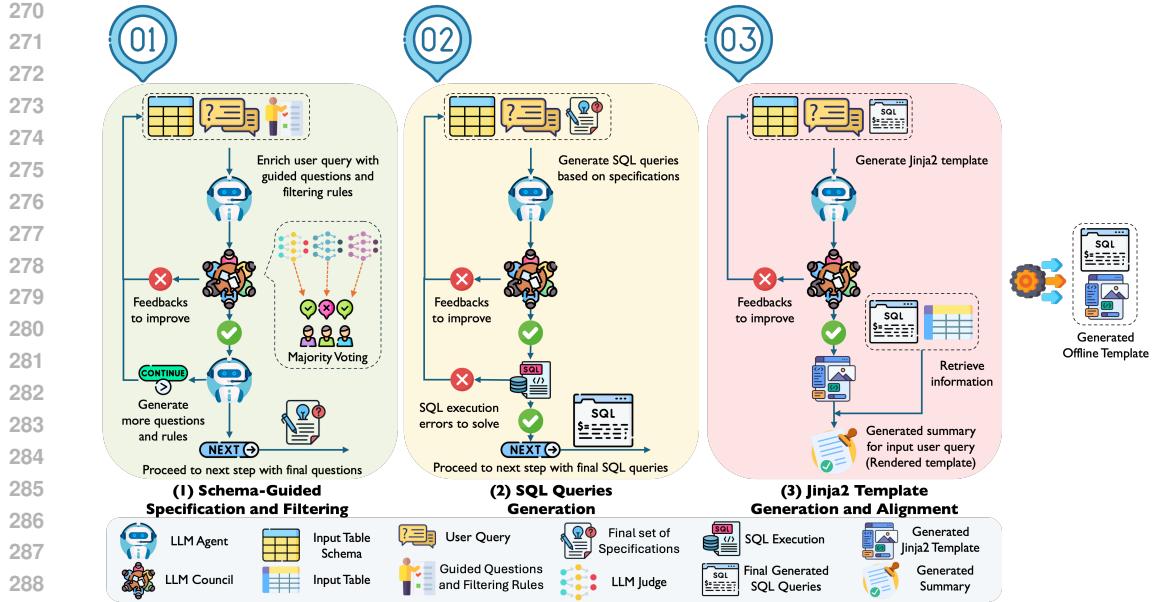


Figure 2: The FACTS framework for query-focused table summarization via Offline Template Generation. (1) **Schema-Guided Specification and Filtering**: the agent enriches the user query with guided questions and filtering rules over the table schema, with validation from the LLM Council. (2) **SQL Queries Generation**: using these specifications, the agent synthesizes and iteratively improves SQL queries through execution feedback and Council validation. (3) **Jinja2 Template Generation and Alignment**: a Jinja2 template verbalizes SQL outputs into natural language, with LLM Council checks ensuring alignment. The final output is a reusable offline template that combines validated SQL queries with a Jinja2 template.

constraints, ensuring that only relevant subsets of the data are processed. Each query is executed locally against the relevant tables to verify correctness. If a query fails or returns empty results, the error traces and execution outputs are passed to the LLM Council for feedback. Based on this feedback, the agent revises the query iteratively until it is executable. This refinement loop ensures that the final SQL queries are robust, accurate, and faithfully grounded in the user specification.

**Stage 3: Jinja2 Template Generation and Alignment.** Once the SQL queries are validated, the agent produces a Jinja2 template to render the results into natural language. The template is required to reference exact column names, correctly iterate over the returned rows, and handle empty results gracefully. The LLM Council then checks for alignment between SQL outputs and template references. If mismatches occur, e.g., missing fields or shape incompatibilities, the SQL and template are refined together until a consistent and valid pair is obtained. The final output is an offline template, consisting of reusable SQL queries and a Jinja2 template that can generalize across new tables with the same schema and query semantics.

Together, these three stages ensure FACTS achieves its key desired properties. Offline templates provide fast summarization by reusing validated SQL queries and Jinja2 template rendering logic, accurate outputs by grounding summaries in executed SQL queries rather than free-form generation, and privacy-compliant operation by exposing only schemas, without revealing raw table values. For completeness, the full prompts used in the FACTS framework are provided in Appendix A.3.

## 4 EXPERIMENTAL RESULTS

In this section, we present a comprehensive evaluation of the proposed FACTS framework. Our experiments are designed to address the following research questions: **RQ1:** Does FACTS, through offline templates, outperform existing methods for query-focused table summarization? **RQ2:** How does FACTS compare with non-agentic alternatives, such as directly prompting an LLM to generate an offline template in a single step? **RQ3:** To what extent does FACTS provide practical benefits in reusability and scalability, particularly when the schema and user query remain fixed or semantically

324 similar, or when table sizes increase? **RQ4: Do human evaluators confirm that FACTS produces**

325 more factually correct and complete summaries with fewer hallucinations than existing baselines?

326 To answer these questions, we first introduce the datasets, evaluation metrics in Section 4.1, and  
 327 baseline methods in Section 4.2. We then provide implementation details for FACTS and all baselines  
 328 in Section 4.3, present the main results and analysis in Section 4.4, conduct ablation studies contrasting  
 329 agentic versus single-call template generation in Section 4.5, and finally evaluate reusability and  
 330 scalability in Section 4.6.

#### 331 4.1 DATASET AND EVALUATION

332 **Datasets.** We evaluate FACTS on the test splits of three widely used benchmarks: FeTaQA (Nan  
 333 et al., 2022), QTSumm (Zhao et al., 2023), and QFMTS (Zhang et al., 2024). FeTaQA consists  
 334 of 2,003 examples from Wikipedia, each pairing a single relational table with a query and a short  
 335 factual summary. QTSumm, also derived from Wikipedia, includes 1,078 examples where queries are  
 336 linked to single tables but require generating longer, paragraph-style summaries. QFMTS contains  
 337 608 examples, with each query associated with an average of 1.8 tables, demanding reasoning and  
 338 integration across multiple table schemas. **QFMTS is based on the Spider dataset (Yu et al., 2018),**  
 339 **which includes 200 databases spanning 138 distinct domains, such as university courses, online**  
 340 **SQL tutorials, textbook examples, and public CSV repositories.** Together, these datasets provide a  
 341 complementary testbed: FeTaQA evaluates concise summarization, QTSumm emphasizes extended  
 342 narrative responses, and QFMTS challenges systems with compositional multi-table reasoning.

342 **Evaluation Metrics.** We assess summarization quality using three standard automatic metrics.  
 343 BLEU (Papineni et al., 2002) measures n-gram precision by computing exact word overlap between  
 344 generated and reference summaries; we report SacreBLEU scores. ROUGE-L (Lin & Hovy, 2003)  
 345 evaluates recall via the longest common subsequence, indicating how much reference content is  
 346 covered; we report the F1 variant. METEOR (Banerjee & Lavie, 2005) balances precision and recall  
 347 by considering unigram matches with stemming and synonymy. Together, these metrics provide a  
 348 comprehensive assessment of both fluency and factual alignment in generated summaries.

#### 349 4.2 BASELINE METHODS

350 We restrict our comparisons to training-free and fine-tuning-free approaches, since FACTS itself does  
 351 not rely on supervised model adaptation. The baselines fall into two categories: prompt-based models  
 352 and agentic frameworks. Prompt-based models include: **(1) Chain-of-Thought (CoT)** (Wei et al.,  
 353 2022) prompts the LLM to explicitly verbalize intermediate reasoning steps before producing the final  
 354 summary. **(2) DirectSumm** (Zhang et al., 2024) generates summaries in a single pass, conditioning  
 355 directly on the table and user query. **(3) ReFactor** (Zhao et al., 2023) extracts query-relevant facts  
 356 from the table and concatenates them with the user query as augmented input to the LLM. **(4) Reason-**  
 357 **then-Summ** (Zhang et al., 2024) decomposes the process into two stages: first retrieving relevant  
 358 facts, then composing a longer narrative summary. Agentic frameworks include: **(5) Binder** (Cheng  
 359 et al., 2023) translates the query into executable SQL programs to ground the results in computation.  
 360 **(6) Dater** (Ye et al., 2023) decomposes large tables into smaller ones and complex queries into  
 361 simpler sub-queries, executes them individually, and aggregates their outputs. **(7) TaPERA** (Zhao  
 362 et al., 2024) generates natural language plans that are converted into Python programs for execution  
 363 and aggregation. **(8) SPaGe** (Zhang et al., 2025) introduces structured graph-based plans, improving  
 364 reliability in multi-table scenarios. Together, these baselines cover the spectrum of training-free  
 365 methods: (i) direct prompting of LLMs with or without explicit reasoning, and (ii) agentic approaches  
 366 that couple LLMs with external executors.

#### 367 4.3 IMPLEMENTATION DETAILS

368 The main LLM agent employs GPT-4o-mini as the backbone model, chosen for its strong performance  
 369 in table reasoning and summarization tasks (Nguyen et al., 2025; Zhang et al., 2025). To further  
 370 align outputs with target writing style, we employ in-context learning (Brown et al., 2020): when  
 371 generating Jinja2 templates, the prompt includes three demonstration examples drawn from the corpus,  
 372 encouraging summaries that are stylistically and structurally consistent with reference outputs.

373 At the workflow level, we allow a fixed upper bound of 10 guided questions and filtering rules, and  
 374 set the patience for revision at 3 iterations for guided questions, filtering rules, SQL queries, and  
 375 Jinja2 templates. SQL execution is handled by DuckDB (Raasveldt & Mühleisen, 2019), which

378  
 379 Table 2: Evaluations on the test sets of three benchmarks. FeTaQA and QTSumm are single-table  
 380 datasets, while QFMTS is a multi-table dataset. The best and second-best results are shown in **bold**  
 381 and underline, respectively. FACTS achieves the best or the second-best results on all datasets.  
 382

Method	FeTaQA			QTSumm			QFMTS		
	BLEU	ROUGE-L	METEOR	BLEU	ROUGE-L	METEOR	BLEU	ROUGE-L	METEOR
CoT	28.2	51.0	56.9	19.3	39.0	47.2	31.5	54.3	58.1
ReFactor	26.2	53.6	57.2	19.9	39.5	48.8	-	-	-
DirectSumm	29.8	51.7	58.2	20.7	40.2	50.3	33.6	57.0	62.8
Reason-then-Summ	31.7	52.6	60.7	<u>21.8</u>	42.3	<b>51.5</b>	40.8	62.7	66.2
Binder	25.5	47.9	51.1	18.2	40.0	39.0	42.5	65.3	70.7
Dater	29.8	54.0	59.4	16.6	35.2	35.5	-	-	-
TaPERA	29.5	53.4	58.2	14.6	33.0	33.2	-	-	-
SPaGe	<b>33.8</b>	55.7	62.3	20.9	41.3	47.7	<u>45.7</u>	68.3	<b>73.4</b>
<b>FACTS (GPT-Only) (ours)</b>	30.8	<u>55.7</u>	<u>66.0</u>	20.1	<u>43.1</u>	50.5	45.4	70.5	<u>73.2</u>
<b>FACTS (ours)</b>	32.6	<b>58.9</b>	<b>67.7</b>	<b>21.9</b>	<b>45.8</b>	<u>51.3</u>	<b>46.0</b>	<b>70.8</b>	<u>73.2</u>

391 enables efficient in-memory querying and integrates seamlessly with pandas DataFrames in Python.  
 392 The LLM Council consists of GPT-4o-mini, Claude-4 Sonnet, and DeepSeek v3. **To further isolate**  
 393 **the impact of Council composition, we also evaluate a FACTS (GPT-Only) variant, in which all**  
 394 **three models in the Council are replaced with GPT-4o-mini, enabling us to assess the effectiveness of**  
 395 **FACTS independent of cross-model diversity.**

396 For baseline methods **and other hyperparameters**, we follow the setup of Zhang et al. (2025). All  
 397 prompt-based and agentic baselines are implemented using the same GPT-4o-mini backbone to ensure  
 398 comparability, and we directly cite reported results from Zhang et al. (2025) where available.

#### 400 4.4 RESULTS AND ANALYSIS

401 **Effectiveness.** Table 2 reports results on the test splits of FeTaQA, QTSumm, and QFMTS. Overall,  
 402 FACTS consistently achieves the best or second-best performance across all datasets and metrics,  
 403 demonstrating the effectiveness of offline template generation with iterative validation. When  
 404 compared with prompt-based methods, FACTS outperforms CoT, ReFactor, and DirectSumm across  
 405 most metrics. These approaches lack grounding in executable programs, which makes them prone  
 406 to hallucinations and incomplete coverage. Reason-then-Summ achieves relatively strong results  
 407 on QTSumm, showing that explicitly structuring the generation process into fact retrieval and  
 408 composition can sometimes improve the quality of the generated summaries. However, its gains are  
 409 inconsistent across datasets, and like other prompt-based models, it lacks execution-level validation  
 410 and remains vulnerable to factual errors and hallucinations in intermediate steps that may propagate  
 411 into the final summary. Against agentic frameworks, FACTS surpasses Binder, Dater, and TaPERA,  
 412 which often struggle with complex logic or multi-table reasoning. SPaGe remains a strong competitor  
 413 by leveraging graph-based planning. Nevertheless, FACTS outperforms SPaGe on every dataset in at  
 414 least two of the reported metrics, suggesting that FACTS generates more faithful and well-formed  
 415 summaries that are better aligned with reference outputs. **We further compare the full FACTS system**  
 416 **with its GPT-Only variant, in which all three models in the LLM Council are replaced by GPT-4o-mini.**  
 417 **While the full FACTS framework achieves the best overall performance, which we attribute in part to**  
 418 **the diversity of reasoning behaviors introduced by heterogeneous Council members, the GPT-Only**  
 419 **variant remains competitive and still outperforms baseline methods on most datasets and metrics.**  
 420 **This result demonstrates that the core FACTS workflow itself is effective even without cross-model**  
 421 **diversity, and that Council heterogeneity further amplifies these strengths.**

421 **Computation Cost.** Across the three datasets evaluated, each sample involves on average 2.47  
 422 accepted guiding questions or filtering rules (2.25 initially accepted and 0.22 accepted after one round  
 423 of revision), 1.36 SQL refinement rounds, and 1.84 template refinement rounds, with the maximum  
 424 patience set to three rounds in our experiments. We further provide token-level efficiency analysis  
 425 showing that the entire FACTS workflow requires 9,922 input tokens and 1,045 output tokens per  
 426 sample on average (including all stages and Council outputs), offering a comprehensive view of  
 427 runtime and generation cost.

428 Taken together, these findings provide a clear answer to **RQ1**, showing that FACTS reliably produces  
 429 offline templates that deliver strong and stable performance across both single-table and multi-table  
 430 summarization tasks, benefiting from the three interconnected stages introduced in Section 3.3. For  
 431 additional qualitative insight, Appendix A.4 provides a step-by-step case study of FACTS, including  
 432 intermediate outputs at each stage.

Table 3: Evaluations of Single-Call on the test sets of three benchmarks.

Method	FeTaQA			QTSumm			QFMTS			Pass Rate
	BLEU	ROUGE-L	METEOR	BLEU	ROUGE-L	METEOR	BLEU	ROUGE-L	METEOR	
Single-Call	29.4	52.1	58.4	14.2	37.9	40.6	35.4	63.2	69.8	83.2%
FACTS (ours)	<b>32.6</b>	<b>58.9</b>	<b>67.7</b>	<b>21.9</b>	<b>45.8</b>	<b>51.3</b>	<b>46.0</b>	<b>70.8</b>	<b>73.2</b>	<b>100.0%</b>

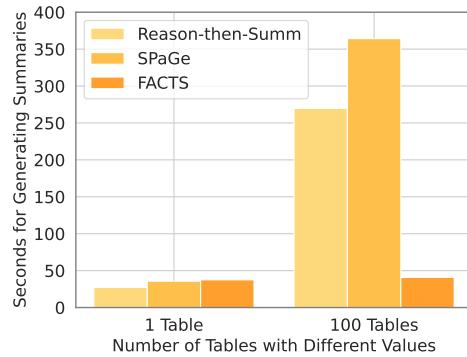


Figure 3: Reusability analysis. Runtime for generating summaries with 1 versus 100 tables under the same schema and query.

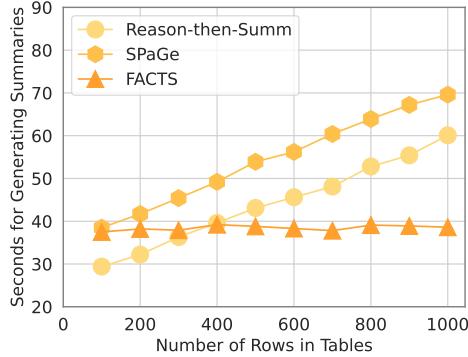


Figure 4: Scalability analysis. Runtime for generating summaries as the number of rows in each table increases.

#### 4.5 ABLATION STUDY

To better assess the role of iterative refinement, we compare FACTS with a simplified *Single-Call* variant. In this setting, the user query, table schema, and three in-context demonstration examples, identical to those used in our main experiments, are provided to GPT-4o-mini, which is prompted to generate an entire offline template in a single step, including both SQL queries and the Jinja2 template. Unlike FACTS, this approach does not incorporate iterative validation or feedback from the LLM Council, nor does it leverage local SQL execution traces during refinement. To capture robustness, we report the SQL *pass rate*, defined as the proportion of generated SQL queries that execute successfully without error.

Results are shown in Table 3. While Single-Call attains moderate text-level scores, it suffers from a substantially lower pass rate. This illustrates the brittleness of one-shot template generation: SQL queries often contain syntax errors, reference non-existent columns, or yield empty outputs, which directly undermines summary quality. By contrast, FACTS consistently achieves a 100% pass rate across datasets, as its iterative refinement loop with Council validation and execution feedback detects and corrects errors before finalization. This not only ensures robustness but also translates into consistently higher BLEU, ROUGE-L, and METEOR scores. In summary, these results directly address **RQ2**, confirming that FACTS substantially outperforms non-agentic single-step alternatives by combining structured stages with iterative validation.

#### 4.6 REUSABILITY AND SCALABILITY ANALYSIS

Finally, we examine whether FACTS delivers the promised advantages of reusability and scalability, addressing **RQ3**. We compare against two strong baselines, Reason-then-Summ and SPaGe, using 100 randomly sampled examples from QTSumm.

**Experiment 1: Reusability across tables.** We fix the user query and table schema but vary the cell values. As shown in Figure 3, with a single table, FACTS is slightly slower than Reason-then-Summ, since it must generate the offline template for the first time, and comparable to SPaGe. **We emphasize that the latency reported in Figure 3 already includes the cost of the initial offline template generation.** However, once the template is generated, FACTS achieves a substantial speed advantage when reusing it across multiple tables with the same schema. With 100 tables under the same schema, FACTS dramatically outperforms both baselines: new summaries require only SQL execution and Jinja2 rendering, while the other methods must reprocess the entire table for every example.

**Experiment 2: Reusability across semantically similar queries.** To assess robustness to semantically similar user queries, we conduct an additional experiment on the same 100 randomly sampled QTSumm examples. We use GPT-5 to paraphrase both the user queries and corresponding refer-

ence summaries, creating semantically similar but lexically different variants. The original offline templates are then applied without regeneration. FACTS maintains comparable performance with  $\text{BLEU} = 21.8$ ,  $\text{ROUGE-L} = 43.5$ , and  $\text{METEOR} = 50.8$ , confirming that it generalizes robustly to semantic variations of the same query while preserving effectiveness.

**Experiment 3: Scalability with table size.** We next test how runtime scales as the number of rows in each table increases, ranging from 100 to 1000. Figure 4 shows that FACTS remains flat in runtime, as templates depend only on the schema. By contrast, Reason-then-Summ and SPaGe incur steadily increasing cost, since larger tables must be serialized and passed into the LLM.

Together, these results show that FACTS achieves both reusability and scalability, while preserving summary quality. This combination of speed, reliability, and accuracy makes it particularly well-suited for real-world deployments.

#### 4.7 HUMAN EVALUATION AND HUMAN PREFERENCE STUDY

To complement the automatic and computational evaluations, we perform human assessments.

**Human Evaluation.** We conduct a comprehensive human evaluation on 100 randomly sampled examples from QTSumm and 100 from QFMTS to assess four aspects: (1) whether each generated SQL semantically matches the user query (intent match), (2) whether the SQL execution results correctly correspond to the numerical or factual content in the reference summary (SQL execution accuracy), (3) whether the numbers and facts rendered in the final summary faithfully reflect the SQL execution results (template rendering accuracy), and (4) whether the LLM Council unanimously accepts a specification or SQL query that leads to an incorrect result (Council consensus error rate). FACTS achieves 97% intent match, 94% SQL execution accuracy, and 98% template rendering accuracy, with a very low Council consensus error rate of about 3%. The Council consensus error rate is computed across two stages: (i) schema-guided specification and filtering, and (ii) SQL query generation. While no errors occur during the specification stage, about 6% of the SQL queries approved by the Council lead to incorrect results during the generation stage, yielding an overall average error rate of approximately 3%. The SQL execution accuracy is lower than the template rendering accuracy because some SQL queries compute incorrect values with respect to the reference summary, while the rendering accuracy reflects whether the template correctly verbalizes the SQL execution results. Therefore, the overall factual correctness of the generated summaries can be estimated as  $94\% \times 98\% \approx 92\%$ . Although FACTS achieves high factual accuracy, the automatic metrics primarily capture surface-level overlap and semantic similarity rather than factual correctness. Human evaluation thus provides complementary insights beyond what automatic metrics can measure.

**Human Preference Study.** We further conduct a side-by-side human preference study comparing FACTS with the strongest baseline SPaGe on QFMTS. Human evaluators are presented with the same user query and two randomly ordered system outputs, one from FACTS and one from SPaGe, without method identifiers. Evaluators are then asked to choose the preferred output or indicate no preference based on three criteria: (1) whether the summary fully answers the user query (completeness), (2) whether the reported numbers and facts are accurate (correctness), and (3) whether unsupported or ungrounded content is introduced (hallucination). FACTS is preferred in 55% of cases for completeness, 59% for correctness, and 60% for hallucination reduction, indicating that human evaluators consistently favor FACTS for producing more accurate, complete, and faithful summaries. These findings confirm that human judgments align with the automatic metrics and computational analyses, collectively addressing **RQ4**.

## 5 CONCLUSION

In this work, we address the challenges in query-focused table summarization by proposing **FACTS**, an agentic framework that generates reusable offline templates by combining schema-guided specifications, SQL synthesis, and Jinja2 rendering, with iterative validation from an LLM Council. Extensive experiments show that FACTS consistently outperforms strong baselines, while offering unique advantages in reusability, scalability, and privacy compliance. We also acknowledge that FACTS assumes a practical privacy model where only table schemas and queries are shared with external LLMs, while raw values remain local. A detailed discussion of this privacy scope, limitations, and possible extensions is provided in Appendix A.5. These results highlight FACTS as a practical solution for query-focused table summarization in real-world applications.

540 REPRODUCIBILITY STATEMENT  
541542 For reproducibility, Section 4.3 outlines the implementation details of our method, while Appendix A.1  
543 and Appendix A.3 provides the detailed prompts used in our method. We will publicly release the  
544 complete codebase once this paper is accepted.

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 REFERENCES  
595

596 Satanjeev Banerjee and Alon Lavie. METEOR: An automatic metric for MT evaluation with improved  
597 correlation with human judgments. In Jade Goldstein, Alon Lavie, Chin-Yew Lin, and Clare Voss  
598 (eds.), *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for  
599 Machine Translation and/or Summarization*, pp. 65–72, Ann Arbor, Michigan, 2005. Association  
600 for Computational Linguistics.

601 Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal,  
602 Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel  
603 Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler,  
604 Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott  
605 Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya  
606 Sutskever, and Dario Amodei. Language models are few-shot learners. In Hugo Larochelle,  
607 Marc’Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances  
608 in Neural Information Processing Systems 33: Annual Conference on Neural Information Process-  
609 ing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

610 Wenhui Chen, Jianshu Chen, Yu Su, Zhiyu Chen, and William Yang Wang. Logical natural language  
611 generation from open-domain tables. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel  
612 Tetreault (eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational  
613 Linguistics*, pp. 7929–7942, Online, 2020. Association for Computational Linguistics.

614 Zhoujun Cheng, Haoyu Dong, Ran Jia, Pengfei Wu, Shi Han, Fan Cheng, and Dongmei Zhang.  
615 FORTAP: Using formulas for numerical-reasoning-aware table pretraining. In Smaranda Muresan,  
616 Preslav Nakov, and Aline Villavicencio (eds.), *Proceedings of the 60th Annual Meeting of the  
617 Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1150–1166, Dublin,  
618 Ireland, 2022a. Association for Computational Linguistics.

619 Zhoujun Cheng, Haoyu Dong, Zhiruo Wang, Ran Jia, Jiaqi Guo, Yan Gao, Shi Han, Jian-Guang  
620 Lou, and Dongmei Zhang. HiTab: A hierarchical table dataset for question answering and  
621 natural language generation. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio  
622 (eds.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics  
623 (Volume 1: Long Papers)*, pp. 1094–1110, Dublin, Ireland, 2022b. Association for Computational  
624 Linguistics.

625 Zhoujun Cheng, Tianbao Xie, Peng Shi, Chengzu Li, Rahul Nadkarni, Yushi Hu, Caiming Xiong,  
626 Dragomir Radev, Mari Ostendorf, Luke Zettlemoyer, Noah A. Smith, and Tao Yu. Binding  
627 language models in symbolic languages. In *The Eleventh International Conference on Learning  
628 Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

629 Mohit Iyyer, Wen-tau Yih, and Ming-Wei Chang. Search-based neural structured learning for  
630 sequential question answering. In Regina Barzilay and Min-Yen Kan (eds.), *Proceedings of the  
631 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*,  
632 pp. 1821–1831, Vancouver, Canada, 2017. Association for Computational Linguistics.

633 Zhengbao Jiang, Yi Mao, Pengcheng He, Graham Neubig, and Weizhu Chen. OmniTab: Pretraining  
634 with natural and synthetic data for few-shot table-based question answering. In Marine Carpuat,  
635 Marie-Catherine de Marneffe, and Ivan Vladimir Meza Ruiz (eds.), *Proceedings of the 2022 Con-  
636 ference of the North American Chapter of the Association for Computational Linguistics: Human  
637 Language Technologies*, pp. 932–942, Seattle, United States, 2022. Association for Computational  
638 Linguistics.

639 Rémi Lebret, David Grangier, and Michael Auli. Neural text generation from structured data  
640 with application to the biography domain. In Jian Su, Kevin Duh, and Xavier Carreras (eds.),  
641 *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp.  
642 1203–1213, Austin, Texas, 2016. Association for Computational Linguistics.

643 Chin-Yew Lin and Eduard Hovy. Automatic evaluation of summaries using n-gram co-occurrence  
644 statistics. In *Proceedings of the 2003 Human Language Technology Conference of the North  
645 American Chapter of the Association for Computational Linguistics*, pp. 150–157, 2003.

648 Ao Liu, Haoyu Dong, Naoaki Okazaki, Shi Han, and Dongmei Zhang. PLOG: Table-to-logic pretrain-  
 649 ing for logical table-to-text generation. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.),  
 650 *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp.  
 651 5531–5546, Abu Dhabi, United Arab Emirates, 2022a. Association for Computational Linguistics.  
 652

653 Qian Liu, Bei Chen, Jiaqi Guo, Morteza Ziyadi, Ziqi Lin, Weizhu Chen, and Jian-Guang Lou. TAPEX:  
 654 table pre-training via learning a neural SQL executor. In *The Tenth International Conference on*  
 655 *Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022b.

656 Nafise Sadat Moosavi, Andreas Rücklé, Dan Roth, and Iryna Gurevych. Scigen: a dataset for  
 657 reasoning-aware text generation from scientific tables. In *Thirty-fifth Conference on Neural*  
 658 *Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

659 Linyong Nan, Chiachun Hsieh, Ziming Mao, Xi Victoria Lin, Neha Verma, Rui Zhang, Wojciech  
 660 Kryściński, Hailey Schoelkopf, Riley Kong, Xiangru Tang, Mutethia Mutuma, Ben Rosand,  
 661 Isabel Trindade, Renusree Bandaru, Jacob Cunningham, Caiming Xiong, Dragomir Radev, and  
 662 Dragomir Radev. FeTaQA: Free-form table question answering. *Transactions of the Association*  
 663 *for Computational Linguistics*, 10:35–49, 2022.

664 Giang Nguyen, Ivan Brugere, Shubham Sharma, Sanjay Kariyappa, Anh Totti Nguyen, and Freddy  
 665 Lecue. Interpretable LLM-based table question answering. *Transactions on Machine Learning*  
 666 *Research*, 2025. ISSN 2835-8856.

667 Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic  
 668 evaluation of machine translation. In Pierre Isabelle, Eugene Charniak, and Dekang Lin (eds.),  
 669 *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pp.  
 670 311–318, Philadelphia, Pennsylvania, USA, 2002. Association for Computational Linguistics.

671 Ankur Parikh, Xuezhi Wang, Sebastian Gehrmann, Manaal Faruqui, Bhuwan Dhingra, Diyi Yang,  
 672 and Dipanjan Das. ToTTo: A controlled table-to-text generation dataset. In Bonnie Webber,  
 673 Trevor Cohn, Yulan He, and Yang Liu (eds.), *Proceedings of the 2020 Conference on Empirical*  
 674 *Methods in Natural Language Processing (EMNLP)*, pp. 1173–1186, Online, 2020. Association  
 675 for Computational Linguistics.

676 Panupong Pasupat and Percy Liang. Compositional semantic parsing on semi-structured tables.  
 677 In Chengqing Zong and Michael Strube (eds.), *Proceedings of the 53rd Annual Meeting of the*  
 678 *Association for Computational Linguistics and the 7th International Joint Conference on Natural*  
 679 *Language Processing (Volume 1: Long Papers)*, pp. 1470–1480, Beijing, China, 2015. Association  
 680 for Computational Linguistics.

681 Mark Raasveldt and Hannes Mühleisen. Duckdb: an embeddable analytical database. In Peter A.  
 682 Boncz, Stefan Manegold, Anastasia Ailamaki, Amol Deshpande, and Tim Kraska (eds.), *Proceed-  
 683 ings of the 2019 International Conference on Management of Data, SIGMOD Conference 2019,  
 684 Amsterdam, The Netherlands, June 30 - July 5, 2019*, pp. 1981–1984. Acm, 2019.

685 Lya Hulliyyatus Suadaa, Hidetaka Kamigaito, Kotaro Funakoshi, Manabu Okumura, and Hiroya  
 686 Takamura. Towards table-to-text generation with numerical reasoning. In Chengqing Zong, Fei Xia,  
 687 Wenjie Li, and Roberto Navigli (eds.), *Proceedings of the 59th Annual Meeting of the Association*  
 688 *for Computational Linguistics and the 11th International Joint Conference on Natural Language*  
 689 *Processing (Volume 1: Long Papers)*, pp. 1451–1465, Online, 2021. Association for Computational  
 690 Linguistics.

691 Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi,  
 692 Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language  
 693 models. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.),  
 694 *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information*  
 695 *Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9,*  
 696 *2022*, 2022.

697 Yunhu Ye, Binyuan Hui, Min Yang, Binhua Li, Fei Huang, and Yongbin Li. Large language models  
 698 are versatile decomposers: Decomposing evidence and questions for table-based reasoning. In  
 699 Hsin-Hsi Chen, Wei-Jou (Edward) Duh, Hen-Hsen Huang, Makoto P. Kato, Josiane Mothe, and

702       Barbara Poblete (eds.), *Proceedings of the 46th International ACM SIGIR Conference on Research*  
 703       *and Development in Information Retrieval, SIGIR 2023, Taipei, Taiwan, July 23-27, 2023*, pp.  
 704       174–184. Acm, 2023.

705       Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene  
 706       Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. Spider: A large-scale  
 707       human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task.  
 708       In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun’ichi Tsujii (eds.), *Proceedings of*  
 709       *the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 3911–3921,  
 710       Brussels, Belgium, October–November 2018. Association for Computational Linguistics. doi:  
 711       10.18653/v1/D18-1425. URL <https://aclanthology.org/D18-1425/>.

712       Weijia Zhang, Vaishali Pal, Jia-Hong Huang, Evangelos Kanoulas, and Maarten de Rijke. Qfmts:  
 713       Generating query-focused summaries over multi-table inputs, 2024.

714       Weijia Zhang, Songgaojun Deng, and Evangelos Kanoulas. Beyond natural language plans: Structure-  
 715       aware planning for query-focused table summarization, 2025.

716       Yilun Zhao, Linyong Nan, Zhenting Qi, Rui Zhang, and Dragomir Radev. ReasTAP: Injecting  
 717       table reasoning skills during pre-training via synthetic reasoning examples. In Yoav Goldberg,  
 718       Zornitsa Kozareva, and Yue Zhang (eds.), *Proceedings of the 2022 Conference on Empirical*  
 719       *Methods in Natural Language Processing*, pp. 9006–9018, Abu Dhabi, United Arab Emirates,  
 720       2022. Association for Computational Linguistics.

721       Yilun Zhao, Zhenting Qi, Linyong Nan, Boyu Mi, Yixin Liu, Weijin Zou, Simeng Han, Ruizhe Chen,  
 722       Xiangru Tang, Yumo Xu, Dragomir Radev, and Arman Cohan. QTSumm: Query-focused sum-  
 723       marization over tabular data. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings*  
 724       *of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 1157–1172,  
 725       Singapore, 2023. Association for Computational Linguistics.

726       Yilun Zhao, Lyuhao Chen, Arman Cohan, and Chen Zhao. TaPERA: Enhancing faithfulness and  
 727       interpretability in long-form table QA by content planning and execution-based reasoning. In  
 728       Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting*  
 729       *of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12824–12840,  
 730       Bangkok, Thailand, 2024. Association for Computational Linguistics.

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 LLM USAGE  
757758 We made use of large language models solely for improving the presentation of this paper. Their  
759 role was limited to refining wording and verifying grammar to enhance clarity and readability. No  
760 assistance from LLMs was involved in the design of methods, implementation of experiments, or  
761 analysis of results.

762

763 A APPENDIX  
764765 A.1 LLM COUNCIL PROMPTS  
766767 For completeness, we include the full prompts used by the LLM Council for evaluation. These  
768 prompts directly correspond to the four validation steps described in Section 3.2 and formalized  
769 in Algorithm 1. Each prompt is presented independently to all LLMs in the Council, and their  
770 structured outputs are aggregated via majority voting with feedback consolidation. We provide the  
771 exact versions here to ensure transparency and reproducibility of our experiments.

772

## 773 Example 2: Prompt for evaluating guided questions and filtering rules.

```

774 You are evaluating a question or filtering rule for table summarization.
775
776 Table Information:
777 [table schema here]
778
779 User Query:
780 [original user query]
781
782 Previously Generated Questions or Filtering Rules:
783 [list of previously accepted guided questions and filtering rules]
784
785 Current Question or Filtering Rule to Evaluate:
786 [proposed guiding question or filtering rule]
787
788 Is this a good question or filtering rule that will help guide SQL query
789 generation? Answer with YES or NO only.
790
791 If NO, provide a brief reason why this question is not helpful.
792
793 Output format:
794 Decision: [YES/NO]
795 Feedback: [Brief reason if NO, or 'Question is good' if YES]
796
797
798
799
800
801
802
803
804
805
806
807
808
809

```

810 Example 3: Prompt for evaluating SQL queries.  
811  
812 You are evaluating a SQL query execution for table summarization.  
813  
814 Table Information:  
815 [table schema here]  
816  
817 Guidance:  
818 [generated guided questions and filtering rules]  
819  
820 SQL Query:  
821 [proposed SQL query]  
822  
823 Execution Result:  
824 [empty results or error message]  
825  
826 Evaluate whether this SQL query is valid and appropriate:  
827 1. Does it execute without errors?  
828 2. Does it return the non-empty data for summarization?  
829 3. Does it filter and select appropriate columns?  
830  
831 Answer with YES or NO only. If NO, provide a brief reason.  
832  
833 Output format:  
834 Decision: [YES/NO]  
835 Feedback: [Brief reason if NO, or 'SQL query is good' if YES]

833 Example 4: Prompt for evaluating SQL-template alignment.  
834  
835 You are evaluating whether a SQL query result aligns with a Jinja2  
836 template for table summarization.  
837  
838 Table Information:  
839 [table schema here]  
840  
841 SQL Query:  
842 [proposed SQL query]  
843  
844 Ninja2 Template:  
845 [proposed Ninja2 template]  
846  
847 Evaluate:  
848 1. Does the SQL return all fields that the template tries to access?  
849 2. Is the data structure compatible (e.g., if template expects multiple  
850 rows, does SQL return them)?  
851 3. Are field names in the template matching the column names returned by  
852 SQL?  
853  
854 Answer with YES or NO only. If NO, provide a brief reason.  
855  
856 Output format:  
857 Decision: [YES/NO]  
858 Feedback: [Brief reason if NO, or 'SQL and template are well-aligned' if  
859 YES]  
860

856  
857  
858  
859  
860  
861  
862  
863

```

864 Example 5: Prompt for evaluating generated summaries.
865
866 You are evaluating a generated summary for table summarization.
867
868 Table Information:
869 [table schema here]
870
871 User Query:
872 [original user query]
873
874 Generated Summary:
875 [system-produced summary]
876
877 Evaluate summary quality:
878 1. Relevance to the query
879 2. Accuracy of information
880 3. Clarity and coherence
881 4. Completeness
882
883 Answer with YES or NO only. If NO, provide a brief reason.
884
885 Output format:
886 Decision: [YES/NO]
887 Feedback: [Brief reason if NO, or 'Summary is good' if YES]
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

```

## A.2 PSEUDO CODE OF FACTS

Algorithm 2 summarizes the FACTS workflow. The process begins with **Schema-Guided Specification and Filtering**, where the agent proposes schema-aware clarifying questions and filtering rules based on the user query  $q$  and table schema  $\mathcal{S}$ . Each specification is vetted by the LLM Council, and accepted ones are accumulated in  $\mathcal{U}$  to progressively refine the query intent. Next, in **SQL Queries Generation**, candidate SQL queries are synthesized using  $(q, \mathcal{U}, \mathcal{S})$ , executed locally against the table, and validated both by execution feedback and the LLM Council. Invalid queries are iteratively revised until a correct and executable query  $Q$  is obtained. Finally, in **Jinja2 Template Generation and Alignment**, the agent generates a Jinja2 template  $\mathcal{J}$  to render SQL results into natural language. The LLM Council checks alignment between template references and SQL outputs; if misalignments are detected, the template is refined until valid. The resulting offline template  $\mathcal{T} = (Q, \mathcal{J})$  can then be reused across any table with the same schema, enabling fast, accurate, and privacy-compliant summarization without exposing raw table values to the LLMs.

## A.3 FACTS PROMPTS

FACTS relies on a set of carefully designed prompts to guide schema-aware question and filtering rule generation, SQL synthesis, and Jinja2 template construction. Below we provide a few representative examples; the complete set of prompts, including dataset-specific variants due to multi-table schemas, is released with our code for reproducibility.

918

**Algorithm 2** FACTS Framework

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

**Input:** user query  $q$ , table schema  $\mathcal{S}$ , LLMs Council  $\mathcal{C}$ , max number of guiding specifications  $K_q$ , patience  $P_q, P_{\text{sql}}, P_{\text{tpl}}$

**Output:** offline template  $\mathcal{T} = (\mathcal{Q}, \mathcal{J})$  where  $\mathcal{Q}$  is a set of SQL queries and  $\mathcal{J}$  is a Jinja2 template

```

1: /* Component 1: Schema-Guided Specification and Filtering */
2:  $\mathcal{U} \leftarrow \emptyset$                                  $\triangleright$  accepted guiding specifications
3: for  $k = 1$  to  $K_q$  do                       $\triangleright$  how many guiding specifications we can generate
4:    $u \leftarrow \text{GENSPECIFICATION}(q, \mathcal{S}, \mathcal{U})$ 
5:   ( $\text{vote}$ ,  $\text{fb}$ )  $\leftarrow \text{COUNCILJUDGE}(\mathcal{C}, u)$ 
6:   if  $\text{vote} = \text{YES}$  then
7:      $\mathcal{U} \leftarrow \mathcal{U} \cup \{u\}$                    $\triangleright$  added to the accepted set of guiding specifications
8:   else
9:      $t \leftarrow 0$ 
10:    while  $\text{vote} \neq \text{YES}$  and  $t < P_q$  do       $\triangleright$  refine until Council satisfied or reach patience
11:       $u \leftarrow \text{REVISESPECIFICATION}(u, \text{fb}, q, \mathcal{S}, \mathcal{U})$ 
12:      ( $\text{vote}$ ,  $\text{fb}$ )  $\leftarrow \text{COUNCILJUDGE}(\mathcal{C}, u)$ 
13:       $t \leftarrow t + 1$ 
14:    if  $\text{vote} = \text{YES}$  then
15:       $\mathcal{U} \leftarrow \mathcal{U} \cup \{u\}$ 
16:    if  $\text{SUFFICIENT}(\mathcal{U})$  then break           $\triangleright$  final set of guiding specifications
17: /* Component 2: SQL Queries Generation */
18:  $\mathcal{Q} \leftarrow \emptyset$ ;  $\text{vote} \leftarrow \text{false}$ 
19:  $t \leftarrow 0$ 
20: while not  $\text{vote}$  and  $t < P_{\text{sql}}$  do
21:    $\tilde{\mathcal{Q}} \leftarrow \text{GENSQL}(q, \mathcal{U}, \mathcal{S})$ 
22:    $\text{exec} \leftarrow \text{EXECUTESQL}(\tilde{\mathcal{Q}}, \mathcal{S})$ 
23:   ( $\text{vote}$ ,  $\text{fb}$ )  $\leftarrow \text{COUNCILJUDGE}(\mathcal{C}, (\tilde{\mathcal{Q}}, \text{exec}))$ 
24:   if  $\text{vote} = \text{YES}$  and  $\text{VALID}(\text{exec})$  then
25:      $\mathcal{Q} \leftarrow \tilde{\mathcal{Q}}$ ;  $\text{vote} \leftarrow \text{true}$ 
26:   else
27:      $\tilde{\mathcal{Q}} \leftarrow \text{REVISESQL}(\tilde{\mathcal{Q}}, \text{exec}, \text{fb})$        $\triangleright$  handle errors, empties, shape mismatches
28:      $t \leftarrow t + 1$ 
29: /* Component 3: Jinja2 Template Generation and Alignment */
30:  $\text{vote} \leftarrow \text{false}$ ;  $t \leftarrow 0$ 
31: while not  $\text{vote}$  and  $t < P_{\text{tpl}}$  do
32:    $\mathcal{J} \leftarrow \text{GENJINJA2}(q, \mathcal{Q}, \mathcal{S})$ 
33:   ( $\text{vote}$ ,  $\text{fb}$ )  $\leftarrow \text{COUNCILJUDGE}(\mathcal{C}, (\mathcal{J}, \mathcal{Q}, \mathcal{S}))$ 
34:    $\text{vote} \leftarrow (\text{vote} = \text{YES}) \text{ and } \text{ALIGNED}(\mathcal{J}, \mathcal{Q}, \mathcal{S})$        $\triangleright$  fields match SQL outputs
35:   if not  $\text{vote}$  then
36:      $(\mathcal{Q}, \mathcal{J}) \leftarrow \text{REFINE}(\mathcal{Q}, \mathcal{J}, \text{fb})$            $\triangleright$  fix unknown fields, shapes
37:      $t \leftarrow t + 1$ 
38: return  $\mathcal{T} = (\mathcal{Q}, \mathcal{J})$                        $\triangleright$  reusable offline template

```

Example 6: Prompt for generating a schema-aware guiding question and filtering rule.

Based on the table information and user query below, generate ONE specific, detailed question or filtering rule that will help guide SQL query generation.

Table Information:  
[table schema here]

User Query: [user query here]

Previously generated questions and filtering rules:  
[None or list of prior questions and filtering rules]

Generate ONE new question or filtering rule that:

```

972 1. Is different from previously generated questions and filtering rules
973 2. Clarifies what specific information is needed or what information is
974     irrelevant
975 3. Helps understand data relationships
976 4. Guides the SQL query structure
977
978 Output format:
979 Specification: [Your single question or filtering rule here]
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025

```

#### Example 7: Prompt for SQL query synthesis.

```

Based on the table information, user query, and refined questions below,
generate a valid DuckDB SQL query.

Table Information:
[table schema here]

Guided Specifications:
[final set of guided questions and filtering rules]

IMPORTANT: You are querying a pandas DataFrame named 'df' that contains
the table data.

Generate valid DuckDB SQL SELECT query that:
1. Retrieves the necessary information to answer the user query
2. Uses proper DuckDB syntax
3. References the DataFrame as 'df'
4. Quotes column names exactly as they appear
5. Handles data types appropriately

Output format:
SQL queries:
[Your SQL query here]

```

#### Example 8: Prompt for Jinja2 template generation.

```

Based on the demonstration examples below and the current SQL result,
generate a Jinja2 template.

--- Demonstration Examples ---
[table, user query, and reference summary triples]

--- Current Task ---
Table Information: [table schema here]
User Query: [user query here]
SQL Query: [SQL query here]

Generate a Jinja2 template that:
1. Uses the variable name 'values' to access the data
2. Iterates with { % for row in values %}
3. Accesses fields with row["Column Name"]
4. Produces a coherent paragraph summary in the style of the examples
5. Handles empty results gracefully

Output format:
Jinja2 template:
[Your Jinja2 template here]

```

For space reasons, we only show these representative prompts here. The full set, including iterative improvement and alignment prompts, is available in our code release.

1026 A.4 CASE STUDY: STEP-BY-STEP OUTPUTS ON QFMTS  
10271028 We illustrate FACTS end-to-end on a QFMTS example, ID #303. The user asks: "Show all document  
1029 names using templates with template type code BK." We show the intermediate artifacts produced at  
1030 each stage.1031 Example 9: Input (QFMTS #303): user query and schemas.  
1032

```

1033 User Query:
1034   Show all document names using templates with template type code BK.
1035 Schemas:
1036   Templates (Template_ID, Version_Number, Template_Type_Code,
1037     Date_Effective_From, Date_Effective_To, Template_Details)
1038   Documents (Document_ID, Template_ID, Document_Name, Document_Description
1039     , Other_Details)

```

1040  
1041 Example 10: Stage 1 — Accepted guided specifications.  
1042

```

1043 Q1: Should results include only documents whose associated BK templates
1044   are currently effective
1045   (i.e., Date_Effective_From <= today and (Date_Effective_To is null or
1046     >= today)),
1047   or should all BK templates be considered regardless of dates?
1048 Q2: Should we return all documents whose Template_ID links to a Templates
1049   row with Template_Type_Code = 'BK'
1050   regardless of the templates' effective date range, or limit results
1051   to BK templates that are currently effective based on
1052     Date_Effective_From and Date_Effective_To?

```

1053 Example 11: Stage 2 — Synthesized SQL (validated by Council).  
1054

```

1055 SELECT d."Document_Name"
1056 FROM "Documents" AS d
1057 JOIN "Templates" AS t
1058   ON d."Template_ID" = t."Template_ID"
1059 WHERE t."Template_Type_Code" = 'BK';

```

1060 Example 12: Stage 3 — Final Jinja2 template (after refinement).  
1061

```

1062 {% set names = (values | map(attribute='Document_Name')) | select() | list
1063   %}
1064 {% set unique_names = names | unique | list %}
1065 {% if unique_names and unique_names|length > 0 %}
1066 There are {{ unique_names|length }} documents that use templates with the
1067   template type code BK.
1068 The document names are {% for n in unique_names %}{{ n }}{% if not loop.
1069   last %}{% if loop.revindex == 1 %}, and {% else %}, {% endif %}{%
1070   endif %}{% endfor %}.
1071 {% else %}
1072 There are 0 documents that use templates with the template type code BK.
1073 {% endif %}

```

**Summary.** The accepted guided specifications focus the retrieval criterion, the synthesized SQL grounds the result set, and the refined Jinja2 template ensures correct counting and list formatting. The final output faithfully matches the reference.

## A.5 PRIVACY SCOPE AND THREAT MODEL

To clarify the privacy assumptions of FACTS, we adopt a practical enterprise-level threat model in which the large language model (LLM) is treated as an external API that can observe schema-level prompts but cannot access any local data or SQL execution results. Raw table cell values (e.g., personal identifiers, transaction records, or numerical measurements) are considered sensitive and never leave the local environment. All interactions with LLMs occur solely at the schema or query level during guided-question generation, SQL synthesis, and template rendering, while SQL execution and summary rendering are performed locally. This design ensures that only structural information, not actual data, is exposed. We acknowledge that schema structures or user queries may still reveal limited information about domain or intent. Future work may integrate stronger defenses such as schema abstraction, name obfuscation, or query redaction to further strengthen privacy protection.