|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| EMPLOYEE\_ID | FIRST\_NAME | LAST\_NAME | EMAIL | PHONE\_NUMBER | HIRE\_DATE | JOB\_ID | SALARY | COMMISSION\_PCT | MANAGER\_ID | DEPARTMENT\_ID |
| 100 | Steven | King | SKING | 515.123.4567 | 2003-06-17 | AD\_PRES | 24000 | 0 | 0 | 90 |
| 101 | Neena | Kochhar | NKOCHHAR | 515.123.4568 | 2005-09-21 | AD\_VP | 17000 | 0 | 100 | 90 |
| 102 | Lex | De Haan | LDEHAAN | 515.123.4569 | 2001-01-13 | AD\_VP | 17000 | 0 | 100 | 90 |
| 103 | Alexander | Hunold | AHUNOLD | 590.423.4567 | 2006-01-03 | IT\_PROG | 9000 | 0 | 102 | 60 |
| 104 | Bruce | Ernst | BERNST | 590.423.4568 | 2007-05-21 | IT\_PROG | 6000 | 0 | 103 | 60 |
| 105 | David | Austin | DAUSTIN | 590.423.4569 | 2005-06-25 | IT\_PROG | 4800 | 0 | 103 | 60 |
| 106 | Valli | Pataballa | VPATABAL | 590.423.4560 | 2006-02-05 | IT\_PROG | 4800 | 0 | 103 | 60 |
| 107 | Diana | Lorentz | DLORENTZ | 590.423.5567 | 2007-02-07 | IT\_PROG | 4200 | 0 | 103 | 60 |
| 108 | Nancy | Greenberg | NGREENBE | 515.124.4569 | 2002-08-17 | FI\_MGR | 12008 | 0 | 101 | 100 |
| 109 | Daniel | Faviet | DFAVIET | 515.124.4169 | 2002-08-16 | FI\_ACCOUNT | 9000 | 0 | 108 | 100 |
| 110 | John | Chen | JCHEN | 515.124.4269 | 2005-09-28 | FI\_ACCOUNT | 8200 | 0 | 108 | 100 |
| 111 | Ismael | Sciarra | ISCIARRA | 515.124.4369 | 2005-09-30 | FI\_ACCOUNT | 7700 | 0 | 108 | 100 |
| 112 | Jose Manuel | Urman | JMURMAN | 515.124.4469 | 2006-03-07 | FI\_ACCOUNT | 7800 | 0 | 108 | 100 |
| 113 | Luis | Popp | LPOPP | 515.124.4567 | 2007-12-07 | FI\_ACCOUNT | 6900 | 0 | 108 | 100 |
| 114 | Den | Raphaely | DRAPHEAL | 515.127.4561 | 2002-12-07 | PU\_MAN | 11000 | 0 | 100 | 30 |
| 115 | Alexander | Khoo | AKHOO | 515.127.4562 | 2003-05-18 | PU\_CLERK | 3100 | 0 | 114 | 30 |
| 116 | Shelli | Baida | SBAIDA | 515.127.4563 | 2005-12-24 | PU\_CLERK | 2900 | 0 | 114 | 30 |
| 117 | Sigal | Tobias | STOBIAS | 515.127.4564 | 2005-07-24 | PU\_CLERK | 2800 | 0 | 114 | 30 |
| 118 | Guy | Himuro | GHIMURO | 515.127.4565 | 2006-11-15 | PU\_CLERK | 2600 | 0 | 114 | 30 |
| 119 | Karen | Colmenares | KCOLMENA | 515.127.4566 | 2007-08-10 | PU\_CLERK | 2500 | 0 | 114 | 30 |
| 120 | Matthew | Weiss | MWEISS | 650.123.1234 | 2004-07-18 | ST\_MAN | 8000 | 0 | 100 | 50 |
| 121 | Adam | Fripp | AFRIPP | 650.123.2234 | 2005-04-10 | ST\_MAN | 8200 | 0 | 100 | 50 |
| 122 | Payam | Kaufling | PKAUFLIN | 650.123.3234 | 2003-05-01 | ST\_MAN | 7900 | 0 | 100 | 50 |
| 123 | Shanta | Vollman | SVOLLMAN | 650.123.4234 | 2005-10-10 | ST\_MAN | 6500 | 0 | 100 | 50 |
| 124 | Kevin | Mourgos | KMOURGOS | 650.123.5234 | 2007-11-16 | ST\_MAN | 5800 | 0 | 100 | 50 |
| 125 | Julia | Nayer | JNAYER | 650.124.1214 | 2005-07-16 | ST\_CLERK | 3200 | 0 | 120 | 50 |
| 126 | Irene | Mikkilineni | IMIKKILI | 650.124.1224 | 2006-09-28 | ST\_CLERK | 2700 | 0 | 120 | 50 |
| 127 | James | Landry | JLANDRY | 650.124.1334 | 2007-01-14 | ST\_CLERK | 2400 | 0 | 120 | 50 |
| 128 | Steven | Markle | SMARKLE | 650.124.1434 | 2008-03-08 | ST\_CLERK | 2200 | 0 | 120 | 50 |
| 129 | Laura | Bissot | LBISSOT | 650.124.5234 | 2005-08-20 | ST\_CLERK | 3300 | 0 | 121 | 50 |
| 130 | Mozhe | Atkinson | MATKINSO | 650.124.6234 | 2005-10-30 | ST\_CLERK | 2800 | 0 | 121 | 50 |
| 131 | James | Marlow | JAMRLOW | 650.124.7234 | 2005-02-16 | ST\_CLERK | 2500 | 0 | 121 | 50 |
| 132 | TJ | Olson | TJOLSON | 650.124.8234 | 2007-04-10 | ST\_CLERK | 2100 | 0 | 121 | 50 |
| 133 | Jason | Mallin | JMALLIN | 650.127.1934 | 2004-06-14 | ST\_CLERK | 3300 | 0 | 122 | 50 |
| 134 | Michael | Rogers | MROGERS | 650.127.1834 | 2006-08-26 | ST\_CLERK | 2900 | 0 | 122 | 50 |
| 135 | Ki | Gee | KGEE | 650.127.1734 | 2007-12-12 | ST\_CLERK | 2400 | 0 | 122 | 50 |
| 136 | Hazel | Philtanker | HPHILTAN | 650.127.1634 | 2008-02-06 | ST\_CLERK | 2200 | 0 | 122 | 50 |
| 137 | Renske | Ladwig | RLADWIG | 650.121.1234 | 2003-07-14 | ST\_CLERK | 3600 | 0 | 123 | 50 |
| 138 | Stephen | Stiles | SSTILES | 650.121.2034 | 2005-10-26 | ST\_CLERK | 3200 | 0 | 123 | 50 |
| 139 | John | Seo | JSEO | 650.121.2019 | 2006-02-12 | ST\_CLERK | 2700 | 0 | 123 | 50 |
| 140 | Joshua | Patel | JPATEL | 650.121.1834 | 2006-04-06 | ST\_CLERK | 2500 | 0 | 123 | 50 |
| 141 | Trenna | Rajs | TRAJS | 650.121.8009 | 2003-10-17 | ST\_CLERK | 3500 | 0 | 124 | 50 |
| 142 | Curtis | Davies | CDAVIES | 650.121.2994 | 2005-01-29 | ST\_CLERK | 3100 | 0 | 124 | 50 |
| 143 | Randall | Matos | RMATOS | 650.121.2874 | 2006-03-15 | ST\_CLERK | 2600 | 0 | 124 | 50 |
| 144 | Peter | Vargas | PVARGAS | 650.121.2004 | 2006-07-09 | ST\_CLERK | 2500 | 0 | 124 | 50 |
| 145 | John | Russell | JRUSSEL | 011.44.1344.429268 | 2004-10-01 | SA\_MAN | 14000 | 0.4 | 100 | 80 |
| 146 | Karen | Partners | KPARTNER | 011.44.1344.467268 | 2005-01-05 | SA\_MAN | 13500 | 0.3 | 100 | 80 |
| 147 | Alberto | Errazuriz | AERRAZUR | 011.44.1344.429278 | 2005-03-10 | SA\_MAN | 12000 | 0.3 | 100 | 80 |
| 148 | Gerald | Cambrault | GCAMBRAU | 011.44.1344.619268 | 2007-10-15 | SA\_MAN | 11000 | 0.3 | 100 | 80 |
| 149 | Eleni | Zlotkey | EZLOTKEY | 011.44.1344.429018 | 2008-01-29 | SA\_MAN | 10500 | 0.2 | 100 | 80 |
| 150 | Peter | Tucker | PTUCKER | 011.44.1344.129268 | 2005-01-30 | SA\_REP | 10000 | 0.3 | 145 | 80 |
| 151 | David | Bernstein | DBERNSTE | 011.44.1344.345268 | 2005-03-24 | SA\_REP | 9500 | 0.25 | 145 | 80 |
| 152 | Peter | Hall | PHALL | 011.44.1344.478968 | 2005-08-20 | SA\_REP | 9000 | 0.25 | 145 | 80 |
| 153 | Christopher | Olsen | COLSEN | 011.44.1344.498718 | 2006-03-30 | SA\_REP | 8000 | 0.2 | 145 | 80 |
| 154 | Nanette | Cambrault | NCAMBRAU | 011.44.1344.987668 | 2006-12-09 | SA\_REP | 7500 | 0.2 | 145 | 80 |
| 155 | Oliver | Tuvault | OTUVAULT | 011.44.1344.486508 | 2007-11-23 | SA\_REP | 7000 | 0.15 | 145 | 80 |
| 156 | Janette | King | JKING | 011.44.1345.429268 | 2004-01-30 | SA\_REP | 10000 | 0.35 | 146 | 80 |
| 157 | Patrick | Sully | PSULLY | 011.44.1345.929268 | 2004-03-04 | SA\_REP | 9500 | 0.35 | 146 | 80 |
| 158 | Allan | McEwen | AMCEWEN | 011.44.1345.829268 | 2004-08-01 | SA\_REP | 9000 | 0.35 | 146 | 80 |
| 159 | Lindsey | Smith | LSMITH | 011.44.1345.729268 | 2005-03-10 | SA\_REP | 8000 | 0.3 | 146 | 80 |
| 160 | Louise | Doran | LDORAN | 011.44.1345.629268 | 2005-12-15 | SA\_REP | 7500 | 0.3 | 146 | 80 |
| 161 | Sarath | Sewall | SSEWALL | 011.44.1345.529268 | 2006-11-03 | SA\_REP | 7000 | 0.25 | 146 | 80 |
| 162 | Clara | Vishney | CVISHNEY | 011.44.1346.129268 | 2005-11-11 | SA\_REP | 10500 | 0.25 | 147 | 80 |
| 163 | Danielle | Greene | DGREENE | 011.44.1346.229268 | 2007-03-19 | SA\_REP | 9500 | 0.15 | 147 | 80 |
| 164 | Mattea | Marvins | MMARVINS | 011.44.1346.329268 | 2008-01-24 | SA\_REP | 7200 | 0.1 | 147 | 80 |
| 165 | David | Lee | DLEE | 011.44.1346.529268 | 2008-02-23 | SA\_REP | 6800 | 0.1 | 147 | 80 |
| 166 | Sundar | Ande | SANDE | 011.44.1346.629268 | 2008-03-24 | SA\_REP | 6400 | 0.1 | 147 | 80 |
| 167 | Amit | Banda | ABANDA | 011.44.1346.729268 | 2008-04-21 | SA\_REP | 6200 | 0.1 | 147 | 80 |
| 168 | Lisa | Ozer | LOZER | 011.44.1343.929268 | 2005-03-11 | SA\_REP | 11500 | 0.25 | 148 | 80 |
| 169 | Harrison | Bloom | HBLOOM | 011.44.1343.829268 | 2006-03-23 | SA\_REP | 10000 | 0.2 | 148 | 80 |
| 170 | Tayler | Fox | TFOX | 011.44.1343.729268 | 2006-01-24 | SA\_REP | 9600 | 0.2 | 148 | 80 |
| 171 | William | Smith | WSMITH | 011.44.1343.629268 | 2007-02-23 | SA\_REP | 7400 | 0.15 | 148 | 80 |
| 172 | Elizabeth | Bates | EBATES | 011.44.1343.529268 | 2007-03-24 | SA\_REP | 7300 | 0.15 | 148 | 80 |
| 173 | Sundita | Kumar | SKUMAR | 011.44.1343.329268 | 2008-04-21 | SA\_REP | 6100 | 0.1 | 148 | 80 |
| 174 | Ellen | Abel | EABEL | 011.44.1644.429267 | 2004-05-11 | SA\_REP | 11000 | 0.3 | 149 | 80 |
| 175 | Alyssa | Hutton | AHUTTON | 011.44.1644.429266 | 2005-03-19 | SA\_REP | 8800 | 0.25 | 149 | 80 |
| 176 | Jonathon | Taylor | JTAYLOR | 011.44.1644.429265 | 2006-03-24 | SA\_REP | 8600 | 0.2 | 149 | 80 |
| 177 | Jack | Livingston | JLIVINGS | 011.44.1644.429264 | 2006-04-23 | SA\_REP | 8400 | 0.2 | 149 | 80 |
| 178 | Kimberely | Grant | KGRANT | 011.44.1644.429263 | 2007-05-24 | SA\_REP | 7000 | 0.15 | 149 | 0 |
| 179 | Charles | Johnson | CJOHNSON | 011.44.1644.429262 | 2008-01-04 | SA\_REP | 6200 | 0.1 | 149 | 80 |
| 180 | Winston | Taylor | WTAYLOR | 650.507.9876 | 2006-01-24 | SH\_CLERK | 3200 | 0 | 120 | 50 |
| 181 | Jean | Fleaur | JFLEAUR | 650.507.9877 | 2006-02-23 | SH\_CLERK | 3100 | 0 | 120 | 50 |
| 182 | Martha | Sullivan | MSULLIVA | 650.507.9878 | 2007-06-21 | SH\_CLERK | 2500 | 0 | 120 | 50 |
| 183 | Girard | Geoni | GGEONI | 650.507.9879 | 2008-02-03 | SH\_CLERK | 2800 | 0 | 120 | 50 |
| 184 | Nandita | Sarchand | NSARCHAN | 650.509.1876 | 2004-01-27 | SH\_CLERK | 4200 | 0 | 121 | 50 |
| 185 | Alexis | Bull | ABULL | 650.509.2876 | 2005-02-20 | SH\_CLERK | 4100 | 0 | 121 | 50 |
| 186 | Julia | Dellinger | JDELLING | 650.509.3876 | 2006-06-24 | SH\_CLERK | 3400 | 0 | 121 | 50 |
| 187 | Anthony | Cabrio | ACABRIO | 650.509.4876 | 2007-02-07 | SH\_CLERK | 3000 | 0 | 121 | 50 |
| 188 | Kelly | Chung | KCHUNG | 650.505.1876 | 2005-06-14 | SH\_CLERK | 3800 | 0 | 122 | 50 |
| 189 | Jennifer | Dilly | JDILLY | 650.505.2876 | 2005-08-13 | SH\_CLERK | 3600 | 0 | 122 | 50 |
| 190 | Timothy | Gates | TGATES | 650.505.3876 | 2006-07-11 | SH\_CLERK | 2900 | 0 | 122 | 50 |
| 191 | Randall | Perkins | RPERKINS | 650.505.4876 | 2007-12-19 | SH\_CLERK | 2500 | 0 | 122 | 50 |
| 192 | Sarah | Bell | SBELL | 650.501.1876 | 2004-02-04 | SH\_CLERK | 4000 | 0 | 123 | 50 |
| 193 | Britney | Everett | BEVERETT | 650.501.2876 | 2005-03-03 | SH\_CLERK | 3900 | 0 | 123 | 50 |
| 194 | Samuel | McCain | SMCCAIN | 650.501.3876 | 2006-07-01 | SH\_CLERK | 3200 | 0 | 123 | 50 |
| 195 | Vance | Jones | VJONES | 650.501.4876 | 2007-03-17 | SH\_CLERK | 2800 | 0 | 123 | 50 |
| 196 | Alana | Walsh | AWALSH | 650.507.9811 | 2006-04-24 | SH\_CLERK | 3100 | 0 | 124 | 50 |
| 197 | Kevin | Feeney | KFEENEY | 650.507.9822 | 2006-05-23 | SH\_CLERK | 3000 | 0 | 124 | 50 |
| 198 | Donald | OConnell | DOCONNEL | 650.507.9833 | 2007-06-21 | SH\_CLERK | 2600 | 0 | 124 | 50 |
| 199 | Douglas | Grant | DGRANT | 650.507.9844 | 2008-01-13 | SH\_CLERK | 2600 | 0 | 124 | 50 |
| 200 | Jennifer | Whalen | JWHALEN | 515.123.4444 | 2003-09-17 | AD\_ASST | 4400 | 0 | 101 | 10 |
| 201 | Michael | Hartstein | MHARTSTE | 515.123.5555 | 2004-02-17 | MK\_MAN | 13000 | 0 | 100 | 20 |
| 202 | Pat | Fay | PFAY | 603.123.6666 | 2005-08-17 | MK\_REP | 6000 | 0 | 201 | 20 |
| 203 | Susan | Mavris | SMAVRIS | 515.123.7777 | 2002-06-07 | HR\_REP | 6500 | 0 | 101 | 40 |
| 204 | Hermann | Baer | HBAER | 515.123.8888 | 2002-06-07 | PR\_REP | 10000 | 0 | 101 | 70 |
| 205 | Shelley | Higgins | SHIGGINS | 515.123.8080 | 2002-06-07 | AC\_MGR | 12008 | 0 | 101 | 110 |
| 206 | William | Gietz | WGIETZ | 515.123.8181 | 2002-06-07 | AC\_ACCOUNT | 8300 | 0 | 205 | 110 |

CREATE TABLE employees (

EMPLOYEE\_ID INT PRIMARY KEY,

FIRST\_NAME VARCHAR(50),

LAST\_NAME VARCHAR(50),

EMAIL VARCHAR(50),

PHONE\_NUMBER VARCHAR(20),

HIRE\_DATE DATE,

JOB\_ID VARCHAR(10),

SALARY DECIMAL(8, 2),

COMMISSION\_PCT DECIMAL(2, 2),

MANAGER\_ID INT,

DEPARTMENT\_ID INT

);

INSERT INTO employees (EMPLOYEE\_ID, FIRST\_NAME, LAST\_NAME, EMAIL, PHONE\_NUMBER, HIRE\_DATE, JOB\_ID, SALARY, COMMISSION\_PCT, MANAGER\_ID, DEPARTMENT\_ID) VALUES (100, 'Steven', 'King', 'SKING', '515.123.4567', '2003-06-17', 'AD\_PRES', 24000.00, NULL, NULL, 90), (101, 'Neena', 'Kochhar', 'NKOCHHAR', '515.123.4568', '2005-09-21', 'AD\_VP', 17000.00, NULL, 100, 90), (102, 'Lex', 'De Haan', 'LDEHAAN', '515.123.4569', '2001-01-13', 'AD\_VP', 17000.00, NULL, 100, 90), (103, 'Alexander', 'Hunold', 'AHUNOLD', '590.423.4567', '2006-01-03', 'IT\_PROG', 9000.00, NULL, 102, 60), (104, 'Bruce', 'Ernst', 'BERNST', '590.423.4568', '2007-05-21', 'IT\_PROG', 6000.00, NULL, 103, 60), (105, 'David', 'Austin', 'DAUSTIN', '590.423.4569', '2005-06-25', 'IT\_PROG', 4800.00, NULL, 103, 60), (106, 'Valli', 'Pataballa', 'VPATABAL', '590.423.4560', '2006-02-05', 'IT\_PROG', 4800.00, NULL, 103, 60), (107, 'Diana', 'Lorentz', 'DLORENTZ', '590.423.5567', '2007-02-07', 'IT\_PROG', 4200.00, NULL, 103, 60);

INSERT INTO employees (EMPLOYEE\_ID, FIRST\_NAME, LAST\_NAME, EMAIL, PHONE\_NUMBER, HIRE\_DATE, JOB\_ID, SALARY, COMMISSION\_PCT, MANAGER\_ID, DEPARTMENT\_ID) VALUES (108, 'Nancy', 'Greenberg', 'NGREENBE', '515.124.4569', '2002-08-17', 'FI\_MGR', 12008.00, NULL, 101, 100), (109, 'Daniel', 'Faviet', 'DFAVIET', '515.124.4169', '2002-08-16', 'FI\_ACCOUNT', 9000.00, NULL, 108, 100), (110, 'John', 'Chen', 'JCHEN', '515.124.4269', '2005-09-28', 'FI\_ACCOUNT', 8200.00, NULL, 108, 100), (111, 'Ismael', 'Sciarra', 'ISCIARRA', '515.124.4369', '2005-09-30', 'FI\_ACCOUNT', 7700.00, NULL, 108, 100), (112, 'Jose Manuel', 'Urman', 'JMURMAN', '515.124.4469', '2006-03-07', 'FI\_ACCOUNT', 7800.00, NULL, 108, 100), (113, 'Luis', 'Popp', 'LPOPP', '515.124.4567', '2007-12-07', 'FI\_ACCOUNT', 6900.00, NULL, 108, 100), (114, 'Den', 'Raphaely', 'DRAPHEAL', '515.127.4561', '2002-12-07', 'PU\_MAN', 11000.00, NULL, 100, 30), (115, 'Alexander', 'Khoo', 'AKHOO', '515.127.4562', '2003-05-18', 'PU\_CLERK', 3100.00, NULL, 114, 30);

INSERT INTO employees (EMPLOYEE\_ID, FIRST\_NAME, LAST\_NAME, EMAIL, PHONE\_NUMBER, HIRE\_DATE, JOB\_ID, SALARY, COMMISSION\_PCT, MANAGER\_ID, DEPARTMENT\_ID) VALUES (116, 'Shelli', 'Baida', 'SBAIDA', '515.127.4563', '2005-12-24', 'PU\_CLERK', 2900.00, NULL, 114, 30), (117, 'Sigal', 'Tobias', 'STOBIAS', '515.127.4564', '2005-07-24', 'PU\_CLERK', 2800.00, NULL, 114, 30), (118, 'Guy', 'Himuro', 'GHIMURO', '515.127.4565', '2006-11-15', 'PU\_CLERK', 2600.00, NULL, 114, 30), (119, 'Karen', 'Colmenares', 'KCOLMENA', '515.127.4566', '2007-08-10', 'PU\_CLERK', 2500.00, NULL, 114, 30), (120, 'Matthew', 'Weiss', 'MWEISS', '650.123.1234', '2004-07-18', 'ST\_MAN', 8000.00, NULL, 100, 50), (121, 'Adam', 'Fripp', 'AFRIPP', '650.123.2234', '2005-04-10', 'ST\_MAN', 8200.00, NULL, 100, 50), (122, 'Payam', 'Kaufling', 'PKAUFLIN', '650.123.3234', '2003-05-01', 'ST\_MAN', 7900.00, NULL, 100, 50), (123, 'Shanta', 'Vollman', 'SVOLLMAN', '650.123.4234', '2005-10-10', 'ST\_MAN', 6500.00, NULL, 100, 50);

INSERT INTO employees (EMPLOYEE\_ID, FIRST\_NAME, LAST\_NAME, EMAIL, PHONE\_NUMBER, HIRE\_DATE, JOB\_ID, SALARY, COMMISSION\_PCT, MANAGER\_ID, DEPARTMENT\_ID) VALUES (124, 'Kevin', 'Mourgos', 'KMOURGOS', '650.123.5234', '2007-11-16', 'ST\_MAN', 5800.00, NULL, 100, 50), (125, 'Julia', 'Nayer', 'JNAYER', '650.124.1214', '2005-07-16', 'ST\_CLERK', 3200.00, NULL, 120, 50), (126, 'Irene', 'Mikkilineni', 'IMIKKILI', '650.124.1224', '2006-09-28', 'ST\_CLERK', 2700.00, NULL, 120, 50), (127, 'James', 'Landry', 'JLANDRY', '650.124.1334', '2007-01-14', 'ST\_CLERK', 2400.00, NULL, 120, 50), (128, 'Steven', 'Markle', 'SMARKLE', '650.124.1434', '2008-03-08', 'ST\_CLERK', 2200.00, NULL, 120, 50), (129, 'Laura', 'Bissot', 'LBISSOT', '650.124.5234', '2005-08-20', 'ST\_CLERK', 3300.00, NULL, 121, 50), (130, 'Mozhe', 'Atkinson', 'MATKINSO', '650.124.6234', '2005-10-30', 'ST\_CLERK', 2800.00, NULL, 121, 50), (131, 'James', 'Marlow', 'JAMRLOW', '650.124.7234', '2005-02-16', 'ST\_CLERK', 2500.00, NULL, 121, 50);

INSERT INTO employees (EMPLOYEE\_ID, FIRST\_NAME, LAST\_NAME, EMAIL, PHONE\_NUMBER, HIRE\_DATE, JOB\_ID, SALARY, COMMISSION\_PCT, MANAGER\_ID, DEPARTMENT\_ID) VALUES (132, 'TJ', 'Olson', 'TJOLSON', '650.124.8234', '2007-04-10', 'ST\_CLERK', 2100.00, NULL, 121, 50), (133, 'Jason', 'Mallin', 'JMALLIN', '650.127.1934', '2004-06-14', 'SA\_MAN', 3300.00, 0.40, 100, 80), (134, 'Michael', 'Rogers', 'MROGERS', '650.127.1834', '2006-08-26', 'SA\_MAN', 2900.00, 0.30, 100, 80), (135, 'Ki', 'Gee', 'KGEE', '650.127.1734', '2007-12-12', 'SA\_MAN', 2400.00, 0.25, 100, 80), (136, 'Hazel', 'Philtanker', 'HPHILTAN', '650.127.1634', '2008-02-06', 'SA\_MAN', 2200.00, 0.20, 100, 80), (137, 'Renske', 'Ladwig', 'RLADWIG', '650.121.1234', '2003-07-14', 'SA\_REP', 3600.00, 0.15, 133, 80), (138, 'Stephen', 'Stiles', 'SSTILES', '650.121.2034', '2005-10-26', 'SA\_REP', 3200.00, 0.10, 133, 80), (139, 'John', 'Seo', 'JSEO', '650.121.2019', '2006-02-12', 'SA\_REP', 2700.00, 0.05, 133, 80);

INSERT INTO employees (EMPLOYEE\_ID, FIRST\_NAME, LAST\_NAME, EMAIL, PHONE\_NUMBER, HIRE\_DATE, JOB\_ID, SALARY, COMMISSION\_PCT, MANAGER\_ID, DEPARTMENT\_ID) VALUES (140, 'Joshua', 'Patel', 'JPATEL', '650.121.1834', '2006-04-06', 'SA\_REP', 2500.00, 0.05, 133, 80), (141, 'Trenna', 'Rajs', 'TRAJS', '650.121.1750', '2003-10-17', 'SA\_REP', 2900.00, 0.05, 133, 80), (142, 'Curtis', 'Davies', 'CDAVIES', '650.121.1554', '2005-01-29', 'SA\_REP', 3100.00, 0.10, 134, 80), (143, 'Randall', 'Matos', 'RMATOS', '650.121.1434', '2006-03-15', 'SA\_REP', 2600.00, 0.05, 134, 80), (144, 'Peter', 'Vargas', 'PVARGAS', '650.121.1334', '2006-07-09', 'SA\_REP', 2500.00, 0.05, 134, 80), (145, 'John', 'Russell', 'JRUSSEL', '011.44.1344.429268', '2004-10-01', 'SA\_REP', 14000.00, 0.40, 100, 80), (146, 'Karen', 'Partners', 'KPARTNER', '011.44.1344.429278', '2005-01-05', 'SA\_REP', 13500.00, 0.30, 100, 80), (147, 'Alberto', 'Errazuriz', 'AERRAZUR', '011.44.1344.429288', '2005-03-10', 'SA\_REP', 12000.00, 0.30, 100, 80);

INSERT INTO employees (EMPLOYEE\_ID, FIRST\_NAME, LAST\_NAME, EMAIL, PHONE\_NUMBER, HIRE\_DATE, JOB\_ID, SALARY, COMMISSION\_PCT, MANAGER\_ID, DEPARTMENT\_ID) VALUES (148, 'Gerald', 'Cambrault', 'GCAMBRAU', '011.44.1344.429318', '2007-10-15', 'SA\_REP', 11000.00, 0.30, 100, 80), (149, 'Eleni', 'Zlotkey', 'EZLOTKEY', '011.44.1344.429338', '2008-01-29', 'SA\_REP', 10500.00, 0.20, 100, 80), (150, 'Peter', 'Tucker', 'PTUCKER', '011.44.1344.129268', '2003-01-30', 'SA\_REP', 10000.00, 0.30, 100, 80), (151, 'David', 'Bernstein', 'DBERNSTE', '011.44.1344.129278', '2005-03-24', 'SA\_REP', 9500.00, 0.25, 100, 80), (152, 'Peter', 'Hall', 'PHALL', '011.44.1344.129288', '2005-08-20', 'SA\_REP', 9000.00, 0.25, 100, 80), (153, 'Christopher', 'Olsen', 'COLSEN', '011.44.1344.129298', '2006-03-30', 'SA\_REP', 8000.00, 0.20, 100, 80), (154, 'Nanette', 'Cambrault', 'NCAMBRAU', '011.44.1344.619268', '2006-12-09', 'SA\_REP', 7500.00, 0.20, 100, 80), (155, 'Oliver', 'Tuvault', 'OTUVAULT', '011.44.1344.619278', '2007-11-23', 'SA\_REP', 7000.00, 0.15, 100, 80);

INSERT INTO employees (EMPLOYEE\_ID, FIRST\_NAME, LAST\_NAME, EMAIL, PHONE\_NUMBER, HIRE\_DATE, JOB\_ID, SALARY, COMMISSION\_PCT, MANAGER\_ID, DEPARTMENT\_ID) VALUES (156, 'Janette', 'King', 'JANKING', '011.44.1344.619288', '2004-01-30', 'SA\_REP', 10000.00, 0.35, 100, 80), (157, 'Patrick', 'Sully', 'PSULLY', '011.44.1344.829268', '2004-03-04', 'SA\_REP', 9500.00, 0.35, 100, 80), (158, 'Allan', 'McEwen', 'AMCEWEN', '011.44.1344.829268', '2004-08-01', 'SA\_REP', 9000.00, 0.35, 100, 80), (159, 'Lindsey', 'Smith', 'LSMITH', '011.44.1344.829278', '2005-03-10', 'SA\_REP', 8000.00, 0.30, 100, 80), (160, 'Louise', 'Doran', 'LDORAN', '011.44.1344.829288', '2005-12-15', 'SA\_REP', 7500.00, 0.30, 100, 80), (161, 'Sarath', 'Sewall', 'SSEWALL', '011.44.1344.929268', '2006-11-03', 'SA\_REP', 7000.00, 0.25, 100, 80), (162, 'Clara', 'Vishney', 'CVISHNEY', '011.44.1344.529268', '2007-11-11', 'SA\_REP', 6500.00, 0.25, 100, 80), (163, 'Danielle', 'Greene', 'DGREENE', '011.44.1344.629268', '2002-03-19', 'SA\_REP', 6000.00, 0.20, 100, 80);

INSERT INTO employees (EMPLOYEE\_ID, FIRST\_NAME, LAST\_NAME, EMAIL, PHONE\_NUMBER, HIRE\_DATE, JOB\_ID, SALARY, COMMISSION\_PCT, MANAGER\_ID, DEPARTMENT\_ID) VALUES (164, 'Mattea', 'Marvins', 'MMARVINS', '011.44.1344.629278', '2003-05-07', 'SA\_REP', 3600.00, 0.15, 145, 80), (165, 'David', 'Lee', 'DLEE', '011.44.1344.629288', '2005-01-19', 'SA\_REP', 3400.00, 0.15, 145, 80), (166, 'Sundar', 'Ande', 'SANDE', '011.44.1344.629299', '2006-03-24', 'SA\_REP', 3000.00, 0.10, 145, 80), (167, 'Amit', 'Banda', 'ABANDA', '011.44.1344.529299', '2006-05-21', 'SA\_REP', 2900.00, 0.10, 145, 80), (168, 'Lisa', 'Ozer', 'LOZER', '011.44.1344.529302', '2007-03-11', 'SA\_REP', 11500.00, 0.25, 146, 80), (169, 'Harrison', 'Bloom', 'HBLOOM', '011.44.1344.529306', '2006-03-23', 'SA\_REP', 10000.00, 0.20, 146, 80), (170, 'Tayler', 'Fox', 'TFOX', '011.44.1344.529309', '2006-01-24', 'SA\_REP', 9600.00, 0.20, 146, 80), (171, 'William', 'Smith', 'WSMITH', '011.44.1344.519301', '2007-02-23', 'SA\_REP', 7400.00, 0.15, 146, 80);

INSERT INTO employees (EMPLOYEE\_ID, FIRST\_NAME, LAST\_NAME, EMAIL, PHONE\_NUMBER, HIRE\_DATE, JOB\_ID, SALARY, COMMISSION\_PCT, MANAGER\_ID, DEPARTMENT\_ID) VALUES (172, 'Elizabeth', 'Bates', 'EBATES', '011.44.1344.519322', '2007-03-24', 'SA\_REP', 7300.00, 0.15, 146, 80), (173, 'Sundita', 'Kumar', 'SKUMAR', '011.44.1344.519333', '2008-04-21', 'SA\_REP', 6100.00, 0.10, 146, 80), (174, 'Ellen', 'Abel', 'EABEL', '011.44.1343.929268', '2004-05-11', 'SA\_REP', 11000.00, 0.30, 100, 80), (175, 'Alyssa', 'Hutton', 'AHUTTON', '011.44.1343.929278', '2006-03-19', 'SA\_REP', 8800.00, 0.25, 100, 80), (176, 'Jonathon', 'Taylor', 'JTAYLOR', '011.44.1343.929288', '2006-06-24', 'SA\_REP', 8600.00, 0.20, 100, 80), (177, 'Jack', 'Livingston', 'JLIVING', '011.44.1343.929299', '2006-04-23', 'SA\_REP', 8400.00, 0.20, 100, 80), (178, 'Kimberely', 'Grant', 'KGRANT', '011.44.1343.929302', '2007-05-24', 'SA\_REP', 7000.00, 0.15, 100, 80), (179, 'Charles', 'Johnson', 'CJOHNSON', '011.44.1343.929306', '2008-01-04', 'SA\_REP', 6200.00, 0.10, 100, 80);

INSERT INTO employees (EMPLOYEE\_ID, FIRST\_NAME, LAST\_NAME, EMAIL, PHONE\_NUMBER, HIRE\_DATE, JOB\_ID, SALARY, COMMISSION\_PCT, MANAGER\_ID, DEPARTMENT\_ID) VALUES (180, 'Winston', 'Taylor', 'WTAYLOR', '650.507.9876', '2006-01-24', 'SH\_CLERK', 3200.00, NULL, 120, 50), (181, 'Jean', 'Fleaur', 'JFLEAUR', '650.507.9877', '2006-02-23', 'SH\_CLERK', 3100.00, NULL, 120, 50), (182, 'Martha', 'Sullivan', 'MSULLIVA', '650.507.9878', '2007-06-21', 'SH\_CLERK', 2500.00, NULL, 120, 50), (183, 'Girard', 'Geoni', 'GGEONI', '650.507.9879', '2008-02-03', 'SH\_CLERK', 2800.00, NULL, 120, 50), (184, 'Nandita', 'Sarchand', 'NSARCHAN', '650.509.1876', '2004-01-27', 'SH\_CLERK', 4200.00, NULL, 121, 50), (185, 'Alexis', 'Bull', 'ABULL', '650.509.2876', '2005-02-20', 'SH\_CLERK', 4100.00, NULL, 121, 50), (186, 'Julia', 'Dellinger', 'JDELLING', '650.509.3876', '2006-06-24', 'SH\_CLERK', 3400.00, NULL, 121, 50), (187, 'Anthony', 'Cabrio', 'ACABRIO', '650.509.4876', '2007-02-07', 'SH\_CLERK', 3000.00, NULL, 121, 50);

INSERT INTO employees (EMPLOYEE\_ID, FIRST\_NAME, LAST\_NAME, EMAIL, PHONE\_NUMBER, HIRE\_DATE, JOB\_ID, SALARY, COMMISSION\_PCT, MANAGER\_ID, DEPARTMENT\_ID) VALUES (188, 'Kelly', 'Chung', 'KCHUNG', '650.505.1876', '2005-06-14', 'SH\_CLERK', 3800.00, NULL, 122, 50), (189, 'Jennifer', 'Dilly', 'JDILLY', '650.505.2876', '2005-08-13', 'SH\_CLERK', 3600.00, NULL, 122, 50), (190, 'Timothy', 'Gates', 'TGATES', '650.505.3876', '2006-07-11', 'SH\_CLERK', 2900.00, NULL, 122, 50), (191, 'Randall', 'Perkins', 'RPERKINS', '650.505.4876', '2007-12-19', 'SH\_CLERK', 2500.00, NULL, 122, 50), (192, 'Sarah', 'Bell', 'SBELL', '650.501.1876', '2004-02-04', 'SH\_CLERK', 4000.00, NULL, 123, 50), (193, 'Britney', 'Everett', 'BEVERETT', '650.501.2876', '2005-03-03', 'SH\_CLERK', 3900.00, NULL, 123, 50), (194, 'Samuel', 'McCain', 'SMCCAIN', '650.501.3876', '2006-07-01', 'SH\_CLERK', 3200.00, NULL, 123, 50), (195, 'Vance', 'Jones', 'VJONES', '650.501.4876', '2007-03-17', 'SH\_CLERK', 2800.00, NULL, 123, 50);

INSERT INTO employees (EMPLOYEE\_ID, FIRST\_NAME, LAST\_NAME, EMAIL, PHONE\_NUMBER, HIRE\_DATE, JOB\_ID, SALARY, COMMISSION\_PCT, MANAGER\_ID, DEPARTMENT\_ID) VALUES (196, 'Alana', 'Walsh', 'AWALSH', '650.507.9811', '2006-04-24', 'SH\_CLERK', 3100.00, NULL, 124, 50), (197, 'Kevin', 'Feeney', 'KFEENEY', '650.507.9822', '2006-05-23', 'SH\_CLERK', 3000.00, NULL, 124, 50), (198, 'Donald', 'OConnell', 'DOCONNEL', '650.507.9833', '2007-06-21', 'SH\_CLERK', 2600.00, NULL, 124, 50), (199, 'Douglas', 'Grant', 'DGRANT', '650.507.9844', '2008-01-13', 'SH\_CLERK', 2400.00, NULL, 124, 50), (200, 'Jennifer', 'Whalen', 'JWHALEN', '515.123.4444', '2003-09-17', 'AD\_ASST', 4400.00, NULL, 101, 10), (201, 'Michael', 'Hartstein', 'MHARTSTE', '515.123.5555', '2004-02-17', 'MK\_MAN', 13000.00, NULL, 100, 20), (202, 'Pat', 'Fay', 'PFAY', '603.123.6666', '2005-08-17', 'MK\_REP', 6000.00, NULL, 201, 20);

INSERT INTO employees (EMPLOYEE\_ID, FIRST\_NAME, LAST\_NAME, EMAIL, PHONE\_NUMBER, HIRE\_DATE, JOB\_ID, SALARY, COMMISSION\_PCT, MANAGER\_ID, DEPARTMENT\_ID) VALUES (203, 'Susan', 'Mavris', 'SMAVRIS', '515.123.7777', '2002-06-07', 'HR\_REP', 6500.00, NULL, 101, 40), (204, 'Hermann', 'Baer', 'HBAER', '515.123.8888', '2002-06-07', 'PR\_REP', 10000.00, NULL, 101, 70);

INSERT INTO employees (EMPLOYEE\_ID, FIRST\_NAME, LAST\_NAME, EMAIL, PHONE\_NUMBER, HIRE\_DATE, JOB\_ID, SALARY, COMMISSION\_PCT, MANAGER\_ID, DEPARTMENT\_ID) VALUES (205, 'Shelley', 'Higgins', 'SHIGGINS', '515.123.8080', '2002-06-07', 'AC\_MGR', 12000.00, NULL, 101, 110), (206, 'William', 'Gietz', 'WGIETZ', '515.123.8181', '2002-06-07', 'AC\_ACCOUNT', 8300.00, NULL, 205, 110);

1. **From the following table, write a SQL query to find those employees who receive a higher salary than the employee with ID 163. Return first name, last name.**

SELECT first\_name, last\_name

FROM employees

WHERE salary > (SELECT salary FROM employees WHERE employee\_id = 163);

1. From the following table, write a SQL query to find out which employees have the same designation as the employee whose ID is 169. Return first name, last name, department ID and job ID.

SELECT first\_name, last\_name, department\_id, job\_id

FROM employees

WHERE job\_id = (SELECT job\_id FROM employees WHERE employee\_id = 169);

1. **From the following table, write a SQL query to find those employees whose salary matches the lowest salary of any of the departments. Return first name, last name and department ID.**

SELECT first\_name, last\_name, department\_id

FROM employees

WHERE salary IN (

SELECT MIN(salary)

FROM employees

GROUP BY department\_id

);

* To find employees whose salary matches the lowest salary of any department, you can use the following SQL query:

SELECT first\_name, last\_name, department\_id

FROM employees

WHERE salary = (

SELECT MIN(salary)

FROM employees

);

1. **From the following table, write a SQL query to find those employees who earn more than the average salary. Return employee ID, first name, last name.**

SELECT employee\_id, first\_name, last\_name

FROM employees

WHERE salary > (

SELECT AVG(salary)

FROM employees

);

**5.** From the following table, write a SQL query to find those employees who report to that manager whose first name is ‘Payam’. Return first name, last name, employee ID and salary.

**SELECT e.first\_name, e.last\_name, e.employee\_id, e.salary**

**FROM employees e**

**JOIN employees m ON e.manager\_id = m.employee\_id**

**WHERE m.first\_name = 'Payam';**

**OR**

SELECT first\_name, last\_name, employee\_id, salary

-- Filtering rows based on the condition that the 'manager\_id' is equal to the result of a subquery

FROM employees

-- Subquery to retrieve the 'employee\_id' of the manager with the first name 'Payam'

WHERE manager\_id =

(SELECT employee\_id

FROM employees

WHERE first\_name = 'Payam'

);

1. From the following tables, write a SQL query to find all those employees who work in the Finance department. Return department ID, name (first), job ID and department name.

**-- Create the departments table**

**CREATE TABLE departments (**

**DEPARTMENT\_ID INT,**

**DEPARTMENT\_NAME VARCHAR(50),**

**MANAGER\_ID INT,**

**LOCATION\_ID INT,**

**PRIMARY KEY (DEPARTMENT\_ID)**

**);**

**-- Insert records into the departments table**

**INSERT INTO departments (DEPARTMENT\_ID, DEPARTMENT\_NAME, MANAGER\_ID, LOCATION\_ID)**

**VALUES**

**(10, 'Administration', 200, 1700),**

**(20, 'Marketing', 201, 1800),**

**(30, 'Purchasing', 114, 1700),**

**(40, 'Human Resources', 203, 2400),**

**(50, 'Shipping', 121, 1500),**

**(60, 'IT', 103, 1400),**

**(70, 'Public Relations', 204, 2700),**

**(80, 'Sales', 145, 2500),**

**(90, 'Executive', 100, 1700),**

**(100, 'Finance', 108, 1700),**

**(110, 'Accounting', 205, 1700),**

**(120, 'Treasury', 0, 1700),**

**(130, 'Corporate Tax', 0, 1700),**

**(140, 'Control And Credit', 0, 1700),**

**(150, 'Shareholder Services', 0, 1700),**

**(160, 'Benefits', 0, 1700),**

**(170, 'Manufacturing', 0, 1700),**

**(180, 'Construction', 0, 1700),**

**(190, 'Contracting', 0, 1700),**

**(200, 'Operations', 0, 1700),**

**(210, 'IT Support', 0, 1700),**

**(220, 'NOC', 0, 1700),**

**(230, 'IT Helpdesk', 0, 1700),**

**(240, 'Government Sales', 0, 1700),**

**(250, 'Retail Sales', 0, 1700),**

**(260, 'Recruiting', 0, 1700),**

**(270, 'Payroll', 0, 1700);**

**SELECT e.employee\_id, e.first\_name, e.job\_id, d.department\_name**

**FROM employees e**

**JOIN departments d ON e.department\_id = d.department\_id**

**WHERE d.department\_name = 'Finance';**

**OR**

**SELECT employee\_id, first\_name, job\_id,**

**(SELECT department\_name**

**FROM departments**

**WHERE department\_id = e.department\_id) AS department\_name**

**FROM employees e**

**WHERE department\_id = (SELECT department\_id**

**FROM departments**

**WHERE department\_name = 'Finance');**

**OR**

SELECT e.department\_id, e.first\_name, e.job\_id, d.department\_name, e.employee\_id

-- Specifying the tables involved in the query ('employees' as 'e' and 'departments' as 'd')

FROM employees e, departments d

-- Joining the 'employees' and 'departments' tables based on the common 'department\_id'

WHERE e.department\_id = d.department\_id

-- Filtering rows based on the condition that the 'department\_name' is 'Finance'

AND d.department\_name = 'Finance';

1. From the following table, write a SQL query to find the employee whose salary is 3000 and reporting person’s ID is 121. Return all fields.

**SELECT \***

**FROM employees**

**WHERE salary = 3000**

**AND manager\_id = 121;**

**OR**

SELECT \*

-- Filtering rows based on the condition that the combination of 'salary' and 'manager\_id' is equal to the result of a subquery

FROM employees

-- Subquery to find the row where 'salary' is 3000 and 'manager\_id' is 121

WHERE (salary, manager\_id) =

(SELECT 3000, 121);

1. From the following table, write a SQL query to find those employees whose ID matches any of the numbers 134, 159 and 183. Return all the fields.

**SELECT \***

**FROM employees**

**WHERE employee\_id IN (134, 159, 183);**

1. From the following table, write a SQL query to find those employees whose salary is in the range of 1000, and 3000 (Begin and end values have included.). Return all the fields.

**SELECT \***

**FROM employees**

**WHERE salary BETWEEN 1000 AND 3000;**

1. From the following table and write a SQL query to find those employees whose salary falls within the range of the smallest salary and 2500. Return all the fields.

**SELECT \***

**FROM employees**

**WHERE salary >= (SELECT MIN(salary) FROM employees)**

**AND salary <= 2500;**

1. From the following tables, write a SQL query to find those employees who do not work in the departments where managers’ IDs are between 100 and 200 (Begin and end values are included.). Return all the fields of the employeess.

**SELECT \***

**FROM employees**

**WHERE department\_id NOT IN (**

**SELECT department\_id**

**FROM departments**

**WHERE manager\_id BETWEEN 100 AND 200**

**);**

**12.** From the following table, write a SQL query to find those employees who get second-highest salary. Return all the fields of the employees.

**SELECT \***

**FROM employees**

**WHERE salary = (**

**SELECT DISTINCT salary**

**FROM employees**

**ORDER BY salary DESC**

**LIMIT 1 OFFSET 1**

**);**

**OR**

SELECT \*

-- Filtering rows based on the condition that the 'employee\_id' is in the result set of a nested subquery

FROM employees

-- Subquery to find 'employee\_id' values where the 'salary' matches the maximum salary below the overall maximum salary

WHERE employee\_id IN

-- Subquery to find 'employee\_id' values where the 'salary' matches the maximum salary below the overall maximum salary

(SELECT employee\_id

FROM employees

-- Subquery to find the maximum salary below the overall maximum salary in the 'employees' table

WHERE salary =

(SELECT MAX(salary)

FROM employees

WHERE salary <

(SELECT MAX(salary) FROM employees)

)

);

**For Finding Third Largest Salary:**

SELECT \*

FROM employees

WHERE salary = (

SELECT DISTINCT salary

FROM employees

ORDER BY salary DESC

LIMIT 1 OFFSET 2

);

**For Finding Fourth Largest Salary:**

SELECT \*

FROM employees

WHERE salary = (

SELECT DISTINCT salary

FROM employees

ORDER BY salary DESC

LIMIT 1 OFFSET 3

);

**For Finding second min salary:**

SELECT \*

FROM employees

WHERE salary = (

SELECT MIN(salary)

FROM employees

WHERE salary > (

SELECT MIN(salary)

FROM employees

)

);

**13.** From the following tables, write a SQL query to find those employees who work in the same department as ‘Clara’. Exclude all those records where first name is ‘Clara’. Return first name, last name and hire date.

SELECT first\_name, last\_name, hire\_date

FROM employees

WHERE department\_id = (

SELECT department\_id

FROM employees

WHERE first\_name = 'Clara'

)

AND first\_name != 'Clara';

**14.** From the following tables, write a SQL query to find those employees who work in a department where the employee’s first name contains the letter 'T'. Return employee ID, first name and last name.

SELECT employee\_id, first\_name, last\_name

FROM employees

WHERE department\_id IN (

SELECT department\_id

FROM employees

WHERE first\_name LIKE '%T%'

);

**15.** From the following tables, write a SQL query to find those employees who earn more than the average salary and work in the same department as an employee whose first name contains the letter 'J'. Return employee ID, first name and salary.

SELECT employee\_id, first\_name, salary

FROM employees

WHERE salary > (

SELECT AVG(salary)

FROM employees

) AND department\_id IN (

SELECT department\_id

FROM employees

WHERE first\_name LIKE '%J%'

);

**16.** From the following table, write a SQL query to find those employees whose department is located at ‘Toronto’. Return first name, last name, employee ID, job ID.

CREATE TABLE locations (

location\_id INT PRIMARY KEY,

street\_address VARCHAR(255),

postal\_code VARCHAR(20),

city VARCHAR(50),

state\_province VARCHAR(50),

country\_id CHAR(2)

);

INSERT INTO locations (location\_id, street\_address, postal\_code, city, state\_province, country\_id) VALUES

(1000, '1297 Via Cola di Rie', '989', 'Roma', NULL, 'IT'),

(1100, '93091 Calle della Testa', '10934', 'Venice', NULL, 'IT'),

(1200, '2017 Shinjuku-ku', '1689', 'Tokyo', 'Tokyo Prefecture', 'JP'),

(1300, '9450 Kamiya-cho', '6823', 'Hiroshima', NULL, 'JP'),

(1400, '2014 Jabberwocky Rd', '26192', 'Southlake', 'Texas', 'US'),

(1500, '2011 Interiors Blvd', '99236', 'South San Francisco', 'California', 'US'),

(1600, '2007 Zagora St', '50090', 'South Brunswick', 'New Jersey', 'US'),

(1700, '2004 Charade Rd', '98199', 'Seattle', 'Washington', 'US'),

(1800, '147 Spadina Ave', 'M5V 2L7', 'Toronto', 'Ontario', 'CA'),

(1900, '6092 Boxwood St', 'YSW 9T2', 'Whitehorse', 'Yukon', 'CA'),

(2000, '40-5-12 Laogianggen', '190518', 'Beijing', NULL, 'CN'),

(2100, '1298 Vileparle (E)', '490231', 'Bombay', 'Maharashtra', 'IN'),

(2200, '12-98 Victoria Street', '2901', 'Sydney', 'New South Wales', 'AU'),

(2300, '198 Clementi North', '540198', 'Singapore', NULL, 'SG'),

(2400, '8204 Arthur St', '', 'London', NULL, 'UK'),

(2500, 'Magdalen Centre, The Oxford Science Park', 'OX9 9ZB', 'Oxford', 'Oxford', 'UK'),

(2600, '9702 Chester Road', '9629850293', 'Stretford', 'Manchester', 'UK'),

(2700, 'Schwanthalerstr. 7031', '80925', 'Munich', 'Bavaria', 'DE'),

(2800, 'Rua Frei Caneca 1360', '01307-002', 'Sao Paulo', 'Sao Paulo', 'BR'),

(2900, '20 Rue des Corps-Saints', '1730', 'Geneva', 'Geneve', 'CH'),

(3000, 'Murtenstrasse 921', '3095', 'Bern', 'BE', 'CH'),

(3100, 'Pieter Breughelstraat 837', '3029SK', 'Utrecht', 'Utrecht', 'NL'),

(3200, 'Mariano Escobedo 9991', '11932', 'Mexico City', 'Distrito Federal', 'MX');

SELECT e.first\_name, e.last\_name, e.employee\_id, e.job\_id

FROM employees e

JOIN departments d ON e.department\_id = d.department\_id

JOIN locations l ON d.location\_id = l.location\_id

WHERE l.city = 'Toronto';

OR

-- Selecting specific columns (first\_name, last\_name, employee\_id, job\_id) from the 'employees' table

SELECT first\_name, last\_name, employee\_id, job\_id

-- Filtering rows based on the condition that the 'department\_id' matches the result of a nested subquery

FROM employees

-- Subquery to find the 'department\_id' where the location's city is 'Toronto'

WHERE department\_id =

(SELECT department\_id

FROM departments

-- Subquery to find the 'location\_id' where the city is 'Toronto' in the 'locations' table

WHERE location\_id =

(SELECT location\_id

FROM locations

WHERE city ='Toronto')

);

**17.** From the following table, write a SQL query to find those employees whose salary is lower than that of employees whose job title is ‘MK\_MAN’. Return employee ID, first name, last name, job ID.

SELECT employee\_id, first\_name, last\_name, job\_id

FROM employees

WHERE salary < (SELECT MAX(salary) FROM employees WHERE job\_id = 'MK\_MAN');

OR

-- Selecting specific columns (employee\_id, first\_name, last\_name, job\_id) from the 'employees' table

SELECT employee\_id, first\_name, last\_name, job\_id

-- Filtering rows based on the condition that the 'salary' is less than any salary in the result set of a subquery

FROM employees

-- Subquery to find salaries where the 'job\_id' is 'MK\_MAN' in the 'employees' table

WHERE salary < ANY

(SELECT salary

FROM employees

-- Subquery to select salaries where the 'job\_id' is 'MK\_MAN'

WHERE job\_id = 'MK\_MAN'

);

**19.** From the following table, write a SQL query to find those employees whose salary exceeds the salary of all those employees whose job title is "PU\_MAN". Exclude job title ‘PU\_MAN’. Return employee ID, first name, last name, job ID.

SELECT employee\_id, first\_name, last\_name, job\_id

FROM employees

WHERE salary > (SELECT MAX(salary) FROM employees WHERE job\_id = 'PU\_MAN')

AND job\_id != 'PU\_MAN';

OR

-- Selecting specific columns (employee\_id, first\_name, last\_name, job\_id) from the 'employees' table

SELECT employee\_id, first\_name, last\_name, job\_id

-- Filtering rows based on the condition that the 'salary' is greater than all salaries in the result set of a subquery

FROM employees

-- Subquery to find salaries where the 'job\_id' is 'PU\_MAN' in the 'employees' table

WHERE salary > ALL

(SELECT salary

FROM employees

-- Subquery to select salaries where the 'job\_id' is 'PU\_MAN'

WHERE job\_id = 'PU\_MAN'

)

-- Additional condition to exclude rows where the 'job\_id' is 'PU\_MAN' from the result set

AND job\_id <> 'PU\_MAN';

**20.** From the following table, write a SQL query to find those employees whose salaries are higher than the average for all departments. Return employee ID, first name, last name, job ID.

-- Selecting specific columns (employee\_id, first\_name, last\_name, job\_id) from the 'employees' table

SELECT employee\_id, first\_name, last\_name, job\_id

-- Filtering rows based on the condition that the 'salary' is greater than all average salaries in the result set of a subquery

FROM employees

-- Subquery to calculate average salaries grouped by 'department\_id' in the 'employees' table

WHERE salary > ALL

(SELECT AVG(salary)

FROM employees

-- Grouping salaries by 'department\_id'

GROUP BY department\_id

);

**21.** From the following table, write a SQL query to check whether there are any employees with salaries exceeding 3700. Return first name, last name and department ID.

-- Selecting specific columns (first\_name, last\_name, department\_id) from the 'employees' table

SELECT first\_name, last\_name, department\_id

-- Filtering rows based on the condition that there exists a row in the subquery result set

FROM employees

-- Subquery to check if there exists any row where the 'salary' is greater than 3700 in the 'employees' table

WHERE EXISTS

(SELECT \*

FROM employees

-- Condition to check if the 'salary' is greater than 3700

WHERE salary > 3700

);

AND

SELECT first\_name, last\_name, department\_id

FROM employees

WHERE salary > 3700;

## First query will return any employee where there exists at least one row with a salary greater than 3700 in the employees table. In contrast, second query will return all employees with a salary greater than 3700.

First query using EXISTS checks if there's at least one row in the subquery result set where the salary is greater than 3700. If such a row exists, it returns all rows from the main query.

Second query directly filters the rows from the employees table based on the salary condition, returning all employees that meet the criteria.

So, if there are multiple employees with salaries exceeding 3700, first query will still return only one row because it's using EXISTS, while second query will return all such employees.

**22.** From the following table, write a SQL query to calculate total salary of the departments where at least one employee works. Return department ID, total salary.

-- Selecting specific columns (departments.department\_id, result1.total\_amt) from the 'departments' table and a derived table (aliased as result1)

SELECT departments.department\_id, result1.total\_amt

-- Cross join between the 'departments' table and a derived table (result1) to combine their data

FROM departments,

-- Subquery to calculate the total salary for each 'department\_id' in the 'employees' table, aliased as result1

(SELECT employees.department\_id, SUM(employees.salary) total\_amt

FROM employees

-- Grouping the data by 'department\_id' to calculate the sum of salaries

GROUP BY department\_id

) result1

-- Matching rows based on the 'department\_id' between the 'departments' table and the result1 derived table

WHERE result1.department\_id = departments.department\_id;

OR

SELECT department\_id, SUM(salary) AS total\_salary

FROM employees

GROUP BY department\_id;

**23.** Write a query to display the employee id, name ( first name and last name ) and the job id column with a modified title SALESMAN for those employees whose job title is ST\_MAN and DEVELOPER for whose job title is IT\_PROG.

SELECT

employee\_id,

CONCAT(first\_name, ' ', last\_name) AS name,

salary,

CASE

WHEN salary > (SELECT AVG(salary) FROM employees) THEN 'HIGH'

WHEN salary < (SELECT AVG(salary) FROM employees) THEN 'LOW'

END AS SalaryStatus

FROM employees;

SELECT

employee\_id,

CONCAT(first\_name, ' ', last\_name) AS name,

CASE

WHEN job\_id = 'ST\_MAN' THEN 'SALESMAN'

WHEN job\_id = 'IT\_PROG' THEN 'DEVELOPER'

ELSE job\_id

END AS modified\_job\_title,

salary

FROM employees;

OR

-- Selecting specific columns (employee\_id, first\_name, last\_name, designation, salary) from the 'employees' table

SELECT employee\_id, first\_name, last\_name,

-- Using the CASE statement to create a new column 'designation' based on the 'job\_id'

CASE job\_id

WHEN 'ST\_MAN' THEN 'SALESMAN'

WHEN 'IT\_PROG' THEN 'DEVELOPER'

ELSE job\_id

END AS designation,

-- Including the 'salary' column in the result set

salary

-- From the 'employees' table

FROM employees;

**24.** Write a query to display the employee id, name ( first name and last name ), salary and the SalaryStatus column with a title HIGH and LOW respectively for those employees whose salary is more than and less than the average salary of all employees.

-- Selecting specific columns (employee\_id, first\_name, last\_name, salary, SalaryStatus) from the 'employees' table

SELECT employee\_id, first\_name, last\_name, salary,

-- Using the CASE statement to create a new column 'SalaryStatus' based on the comparison of 'salary' with the average salary in the 'employees' table

CASE WHEN salary >= (SELECT AVG(salary) FROM employees) THEN 'HIGH'

ELSE 'LOW'

END AS SalaryStatus

-- From the 'employees' table

FROM employees;

OR

SELECT

employee\_id,

CONCAT(first\_name, ' ', last\_name) AS name,

salary,

CASE

WHEN salary > (SELECT AVG(salary) FROM employees) THEN 'HIGH'

WHEN salary < (SELECT AVG(salary) FROM employees) THEN 'LOW'

END AS SalaryStatus

FROM employees;

**25.** Write a query to display the employee id, name ( first name and last name ), SalaryDrawn, AvgCompare (salary - the average salary of all employees) and the SalaryStatus column with a title HIGH and LOW respectively for those employees whose salary is more than and less than the average salary of all employees.

SELECT

employee\_id,

CONCAT(first\_name, ' ', last\_name) AS name,

salary AS SalaryDrawn,

salary - (SELECT AVG(salary) FROM employees) AS AvgCompare,

CASE

WHEN salary > (SELECT AVG(salary) FROM employees) THEN 'HIGH'

WHEN salary < (SELECT AVG(salary) FROM employees) THEN 'LOW'

END AS SalaryStatus

FROM employees;

OR

-- Selecting specific columns (employee\_id, first\_name, last\_name, salary AS SalaryDrawn, AvgCompare, SalaryStatus) from the 'employees' table

SELECT employee\_id, first\_name, last\_name, salary AS SalaryDrawn,

-- Using the ROUND function to calculate the difference between 'salary' and the average salary in the 'employees' table, aliased as AvgCompare

ROUND((salary - (SELECT AVG(salary) FROM employees)), 2) AS AvgCompare,

-- Using the CASE statement to create a new column 'SalaryStatus' based on the comparison of 'salary' with the average salary in the 'employees' table

CASE WHEN salary >= (SELECT AVG(salary) FROM employees) THEN 'HIGH'

ELSE 'LOW'

END AS SalaryStatus

-- From the 'employees' table

FROM employees;

OR

SELECT

employee\_id,

first\_name,

last\_name,

salary AS SalaryDrawn,

salary - (SELECT AVG(salary) FROM employees) AS AvgCompare,

CASE

WHEN salary >= (SELECT AVG(salary) FROM employees) THEN 'HIGH'

ELSE 'LOW'

END AS SalaryStatus

FROM employees;

**26.** From the following table, write a SQL query to find all those departments where at least one employee is employed. Return department name.

SELECT DISTINCT department\_name

FROM employees

JOIN departments ON employees.department\_id = departments.department\_id;

OR

-- Selecting the 'department\_name' column from the 'departments' table

SELECT department\_name

-- Filtering rows based on the condition that the 'department\_id' is in the result set of a subquery

FROM departments

-- Subquery to find distinct 'department\_id' values from the 'employees' table

WHERE department\_id IN

(SELECT DISTINCT(department\_id)

FROM employees);

**27.** From the following tables, write a SQL query to find employees who work in departments located in the United Kingdom. Return first name.

CREATE TABLE countries (

COUNTRY\_ID VARCHAR(2) PRIMARY KEY,

COUNTRY\_NAME VARCHAR(50),

REGION\_ID INT

);

INSERT INTO countries (COUNTRY\_ID, COUNTRY\_NAME, REGION\_ID) VALUES

('AR', 'Argentina', 2),

('AU', 'Australia', 3),

('BE', 'Belgium', 1),

('BR', 'Brazil', 2),

('CA', 'Canada', 2),

('CH', 'Switzerland', 1),

('CN', 'China', 3),

('DE', 'Germany', 1),

('DK', 'Denmark', 1),

('EG', 'Egypt', 4),

('FR', 'France', 1),

('IL', 'Israel', 4),

('IN', 'India', 3),

('IT', 'Italy', 1),

('JP', 'Japan', 3),

('KW', 'Kuwait', 4),

('ML', 'Malaysia', 3),

('MX', 'Mexico', 2),

('NG', 'Nigeria', 4),

('NL', 'Netherlands', 1),

('SG', 'Singapore', 3),

('UK', 'United Kingdom', 1),

('US', 'United States of America', 2),

('ZM', 'Zambia', 4),

('ZW', 'Zimbabwe', 4);

SELECT first\_name

FROM employees

WHERE department\_id IN (

SELECT department\_id

FROM departments

WHERE location\_id IN (

SELECT location\_id

FROM locations

WHERE country\_id = 'UK'

)

);

**28.** From the following table, write a SQL query to find out which employees are earning more than the average salary and who work in any of the IT departments. Return last name.

SELECT last\_name

FROM employees

WHERE salary > (

SELECT AVG(salary)

FROM employees

)

AND department\_id IN (

SELECT department\_id

FROM departments

WHERE department\_name LIKE '%IT%'

);

**29.** From the following table, write a SQL query to find all those employees who earn more than an employee whose last name is 'Ozer'. Sort the result in ascending order by last name. Return first name, last name and salary.

SELECT first\_name, last\_name, salary

FROM employees

WHERE salary > (

SELECT salary

FROM employees

WHERE last\_name = 'Ozer'

)

ORDER BY last\_name ASC;

**30.** From the following tables, write a SQL query find the employees who report to a manager based in the United States. Return first name, last name.

-- Selecting specific columns (first\_name, last\_name) from the 'employees' table

SELECT first\_name, last\_name

-- Filtering rows based on the condition that the 'manager\_id' is in the result set of a nested subquery

FROM employees

-- Subquery to find 'employee\_id' values where the 'department\_id' is in the result set of another subquery

WHERE manager\_id IN

(SELECT employee\_id

FROM employees

-- Subquery to find 'department\_id' values where the 'location\_id' is in the result set of another subquery

WHERE department\_id IN

-- Subquery to find 'department\_id' values where the 'country\_id' is 'US' in the 'locations' table

(SELECT department\_id

FROM departments

-- Subquery to find 'location\_id' values where the 'country\_id' is 'US' in the 'countries' table

WHERE location\_id IN

(SELECT location\_id

FROM locations

WHERE country\_id='US')

)

);

**31.** From the following tables, write a SQL query to find those employees whose salaries exceed 50% of their department's total salary bill. Return first name, last name.

-- Selecting specific columns (e1.first\_name, e1.last\_name) from the 'employees' table, aliased as e1

SELECT e1.first\_name, e1.last\_name

-- Filtering rows based on the condition that the 'salary' is greater than half the sum of salaries within the same department

FROM employees e1

-- Subquery to calculate half the sum of salaries for each department in the 'employees' table, aliased as e2

WHERE salary >

(SELECT (SUM(salary)) \* 0.5

-- From the 'employees' table, aliased as e2

FROM employees e2

-- Matching rows based on the equality of 'department\_id' between e1 and e2

WHERE e1.department\_id = e2.department\_id

);

OR

SELECT e.first\_name, e.last\_name

FROM employees e

JOIN (

SELECT department\_id, SUM(salary) AS total\_salary

FROM employees

GROUP BY department\_id

) dept\_sal ON e.department\_id = dept\_sal.department\_id

WHERE e.salary > 0.5 \* dept\_sal.total\_salary;

**32.** From the following tables, write a SQL query to find those employees who are managers. Return all the fields of employees table.

-- Selecting all columns (\*) from the 'employees' table

SELECT \*

-- Filtering rows based on the condition that the 'employee\_id' is in the result set of a subquery

FROM employees

-- Subquery to find distinct 'manager\_id' values from the 'employees' table

WHERE employee\_id IN

(SELECT DISTINCT manager\_id FROM employees);

OR

SELECT \*

FROM employees

WHERE employee\_id IN (

SELECT manager\_id

FROM employees

WHERE manager\_id IS NOT NULL

);

**33.** From the following table, write a SQL query to find those employees who manage a department. Return all the fields of employees table.

-- Selecting all columns (\*) from the 'employees' table

SELECT \*

-- Filtering rows based on the condition that the 'employee\_id' is equal to any 'manager\_id' in the result set of a subquery

FROM employees

-- Subquery to find 'manager\_id' values from the 'departments' table

WHERE employee\_id = ANY

(SELECT manager\_id FROM departments );

**34.** From the following table, write a SQL query to search for employees who receive such a salary, which is the maximum salary for salaried employees, hired between January 1st, 2002 and December 31st, 2003. Return employee ID, first name, last name, salary, department name and city.

-- Selecting specific columns (a.employee\_id, a.first\_name, a.last\_name, a.salary, b.department\_name, c.city) from the 'employees' table (aliased as a), 'departments' table (aliased as b), and 'locations' table (aliased as c)

SELECT a.employee\_id, a.first\_name, a.last\_name, a.salary, b.department\_name, c.city

-- Filtering rows based on the condition that the 'salary' of employee a is equal to the maximum salary within a specified date range

FROM employees a, departments b, locations c

-- Subquery to find the maximum 'salary' within the date range '01/01/2002' to '12/31/2003' in the 'employees' table

WHERE a.salary =

(SELECT MAX(salary)

FROM employees

WHERE hire\_date BETWEEN '01/01/2002' AND '12/31/2003')

-- Additional conditions to match 'department\_id' between a and b, and 'location\_id' between b and c

AND a.department\_id=b.department\_id

AND b.location\_id=c.location\_id;

**OR**

SELECT a.employee\_id, a.first\_name, a.last\_name, a.salary, b.department\_name, c.city

FROM employees a

JOIN departments b ON a.department\_id = b.department\_id

JOIN locations c ON b.location\_id = c.location\_id

WHERE a.salary =

(SELECT MAX(salary)

FROM employees

WHERE hire\_date BETWEEN '2002-01-01' AND '2003-12-31');

**35.** From the following tables, write a SQL query to find those departments that are located in the city of London. Return department ID, department name.

-- Selecting specific columns (department\_id, department\_name) from the 'departments' table

SELECT department\_id, department\_name

-- Filtering rows based on the condition that the 'location\_id' matches the result of a subquery

FROM departments

-- Subquery to find the 'location\_id' where the city is 'London' in the 'locations' table

WHERE location\_id =

(SELECT location\_id

FROM locations

-- Condition to find the 'location\_id' where the city is 'London'

WHERE city = 'London'

);

**36.** From the following table, write a SQL query to find those employees who earn more than the average salary. Sort the result-set in descending order by salary. Return first name, last name, salary, and department ID.

-- Selecting specific columns (first\_name, last\_name, salary, department\_id) from the 'employees' table

SELECT first\_name, last\_name, salary, department\_id

-- Filtering rows based on the condition that the 'salary' is greater than the average salary in the 'employees' table

FROM employees

-- Subquery to calculate the average salary from the 'employees' table

WHERE salary > (

SELECT AVG(salary)

FROM employees )

-- Ordering the result set based on the 'salary' column in descending order

ORDER BY salary DESC;

**37.** From the following table, write a SQL query to find those employees who earn more than the maximum salary for a department of ID 40. Return first name, last name and department ID.

-- Selecting specific columns (first\_name, last\_name, salary, department\_id) from the 'employees' table

SELECT first\_name, last\_name, salary, department\_id

-- Filtering rows based on the condition that the 'salary' is greater than all salaries in the result set of a subquery

FROM employees

-- Subquery to find salaries where the 'department\_id' is 40 in the 'employees' table

WHERE salary > ALL

(SELECT salary

FROM employees

WHERE department\_id = 40);

**38.** From the following table, write a SQL query to find departments for a particular location. The location matches the location of the department of ID 30. Return department name and department ID.

-- Selecting specific columns (department\_name, department\_id) from the 'departments' table

SELECT department\_name, department\_id

-- Filtering rows based on the condition that the 'location\_id' matches the result of a subquery

FROM departments

-- Subquery to find the 'location\_id' where the 'department\_id' is 30 in the 'departments' table

WHERE location\_id =

(

SELECT location\_id

FROM departments

WHERE department\_id = 30

);

**39.** From the following table, write a SQL query to find employees who work for the department in which employee ID 201 is employed. Return first name, last name, salary, and department ID.

-- Selecting specific columns (first\_name, last\_name, salary, department\_id) from the 'employees' table

SELECT first\_name, last\_name, salary, department\_id

-- Filtering rows based on the condition that the 'department\_id' matches the result of a subquery

FROM employees

-- Subquery to find the 'department\_id' where the 'employee\_id' is 201 in the 'employees' table

WHERE department\_id =

(SELECT department\_id

FROM employees

WHERE employee\_id = 201);

**40.** From the following table, write a SQL query to find those employees whose salary matches that of the employee who works in department ID 40. Return first name, last name, salary, and department ID.

-- Selecting specific columns (first\_name, last\_name, salary, department\_id) from the 'employees' table

SELECT first\_name, last\_name, salary, department\_id

-- Filtering rows based on the condition that the 'salary' is in the result set of a subquery

FROM employees

-- Subquery to find salaries where the 'department\_id' is 40 in the 'employees' table

WHERE salary IN (

SELECT salary

FROM employees

WHERE department\_id = 40);

**41.** From the following table, write a SQL query to find those employees who work in the department 'Marketing'. Return first name, last name and department ID.

-- Selecting specific columns (first\_name, last\_name, department\_id) from the 'employees' table

SELECT first\_name, last\_name, department\_id

-- Filtering rows based on the condition that the 'department\_id' matches the result of a subquery

FROM employees

-- Subquery to find the 'department\_id' where the 'department\_name' is 'Marketing' in the 'departments' table

WHERE department\_id =

(SELECT department\_id

FROM departments

WHERE department\_name = 'Marketing');

**42.** From the following table, write a SQL query to find those employees who earn more than the minimum salary of a department of ID 40. Return first name, last name, salary, and department ID.

-- Selecting specific columns (first\_name, last\_name, salary, department\_id) from the 'employees' table

SELECT first\_name, last\_name, salary, department\_id

-- Filtering rows based on the condition that the 'salary' is greater than any salary in the result set of a subquery

FROM employees

-- Subquery to find salaries where the 'department\_id' is 40 in the 'employees' table

WHERE salary > ANY

(SELECT salary

FROM employees

WHERE department\_id = 40);

**43.** From the following table, write a SQL query to find those employees who joined after the employee whose ID is 165. Return first name, last name and hire date.

-- Selecting a concatenated string of 'first\_name' and 'last\_name' as 'Full\_Name', and the 'hire\_date' from the 'employees' table

SELECT first\_name ||' '|| last\_name AS Full\_Name , hire\_date

-- Filtering rows based on the condition that the 'hire\_date' is greater than the 'hire\_date' of the employee with 'employee\_id' 165 in the 'employees' table

FROM employees

-- Subquery to find the 'hire\_date' where the 'employee\_id' is 165 in the 'employees' table

WHERE hire\_date > (

SELECT hire\_date

FROM employees

WHERE employee\_id = 165);

**44.** From the following table, write a SQL query to find those employees who earn less than the minimum salary of a department of ID 70. Return first name, last name, salary, and department ID.

-- Selecting specific columns (first\_name, last\_name, salary, department\_id) from the 'employees' table

SELECT first\_name, last\_name, salary, department\_id

-- Filtering rows based on the condition that the 'salary' is less than all salaries in the result set of a subquery

FROM employees

-- Subquery to find salaries where the 'department\_id' is 70 in the 'employees' table

WHERE salary < ALL

(SELECT salary

FROM employees

WHERE department\_id = 70);

**45.** From the following table, write a SQL query to find those employees who earn less than the average salary and work at the department where Laura (first name) is employed. Return first name, last name, salary, and department ID.

-- Selecting specific columns (first\_name, last\_name, salary, department\_id) from the 'employees' table

SELECT first\_name, last\_name, salary, department\_id

-- Filtering rows based on the condition that the 'salary' is less than the average salary in the 'employees' table

FROM employees

-- Subquery to find the average salary from the 'employees' table

WHERE salary <

(SELECT AVG(salary)

FROM employees )

-- Additional condition to filter rows where the 'department\_id' matches the result of a subquery

AND department\_id =

(SELECT department\_id

FROM employees

WHERE first\_name = 'Laura');

**46.** From the following tables, write a SQL query to find all employees whose department is located in London. Return first name, last name, salary, and department ID.

-- Selecting specific columns (first\_name, last\_name, salary, department\_id) from the 'employees' table

SELECT first\_name, last\_name, salary, department\_id

-- Filtering rows based on the condition that the 'department\_id' is in the result set of a nested subquery

FROM employees

-- Subquery to find 'department\_id' values where the 'location\_id' matches the result of another subquery

WHERE department\_id IN

(SELECT department\_id

FROM departments

-- Subquery to find 'location\_id' values where the 'city' is 'London' in the 'locations' table

WHERE location\_id =

(SELECT location\_id

FROM locations

-- Condition to find 'location\_id' values where the 'city' is 'London'

WHERE city = 'London')

);

**47.** From the following tables, write a SQL query to find the city of the employee of ID 134. Return city.

-- Selecting the 'city' column from the 'locations' table

SELECT city

-- Filtering rows based on the condition that the 'location\_id' matches the result of a nested subquery

FROM locations

-- Subquery to find the 'location\_id' where the 'department\_id' matches the result of another subquery in the 'departments' table

WHERE location\_id =

(SELECT location\_id

FROM departments

-- Subquery to find the 'department\_id' where the 'employee\_id' is 134 in the 'employees' table

WHERE department\_id =

(SELECT department\_id

FROM employees

-- Condition to find the 'department\_id' where the 'employee\_id' is 134

WHERE employee\_id=134)

);

**48.** From the following tables, write a SQL query to find those departments where maximum salary is 7000 and above. The employees worked in those departments have already completed one or more jobs. Return all the fields of the departments.

-- Create the job\_history table

CREATE TABLE job\_history (

EMPLOYEE\_ID INT,

START\_DATE DATE,

END\_DATE DATE,

JOB\_ID VARCHAR(20),

DEPARTMENT\_ID INT

);

-- Insert records into the job\_history table

INSERT INTO job\_history (EMPLOYEE\_ID, START\_DATE, END\_DATE, JOB\_ID, DEPARTMENT\_ID) VALUES

(102, '2001-01-13', '2006-07-24', 'IT\_PROG', 60),

(101, '1997-09-21', '2001-10-27', 'AC\_ACCOUNT', 110),

(101, '2001-10-28', '2005-03-15', 'AC\_MGR', 110),

(201, '2004-02-17', '2007-12-19', 'MK\_REP', 20),

(114, '2006-03-24', '2007-12-31', 'ST\_CLERK', 50),

(122, '2007-01-01', '2007-12-31', 'ST\_CLERK', 50),

(200, '1995-09-17', '2001-06-17', 'AD\_ASST', 90),

(176, '2006-03-24', '2006-12-31', 'SA\_REP', 80),

(176, '2007-01-01', '2007-12-31', 'SA\_MAN', 80),

(200, '2002-07-01', '2006-12-31', 'AC\_ACCOUNT', 90);

-- Selecting all columns (\*) from the 'departments' table

SELECT \*

-- Filtering rows based on the condition that the 'DEPARTMENT\_ID' is in the result set of a nested subquery

FROM departments

-- Subquery to find 'DEPARTMENT\_ID' values where the 'EMPLOYEE\_ID' is in the result set of another subquery in the 'employees' table

WHERE DEPARTMENT\_ID IN

(SELECT DEPARTMENT\_ID

FROM employees

-- Subquery to find 'EMPLOYEE\_ID' values where the count of occurrences is greater than 1 in the 'job\_history' table

WHERE EMPLOYEE\_ID IN

(SELECT EMPLOYEE\_ID

FROM job\_history

-- Grouping 'EMPLOYEE\_ID' values and filtering based on the count of occurrences

GROUP BY EMPLOYEE\_ID

HAVING COUNT(EMPLOYEE\_ID) > 1)

-- Grouping 'DEPARTMENT\_ID' values and filtering based on the maximum 'SALARY'

GROUP BY DEPARTMENT\_ID

HAVING MAX(SALARY) > 7000);

**49.** From the following tables, write a SQL query to find those departments where the starting salary is at least 8000. Return all the fields of departments.

-- Selecting all columns (\*) from the 'departments' table

SELECT \*

-- Filtering rows based on the condition that the 'department\_id' is in the result set of a subquery

FROM departments

-- Subquery to find 'department\_id' values where the minimum 'salary' is greater than or equal to 8000 in the 'employees' table

WHERE department\_id IN

( SELECT department\_id

FROM employees

-- Grouping 'department\_id' values and filtering based on the minimum 'salary'

GROUP BY department\_id

HAVING MIN(salary) >= 8000

);

**50.** From the following table, write a SQL query to find those managers who supervise four or more employees. Return manager name, department ID.

-- Selecting a concatenated string of 'first\_name' and 'last\_name' as 'Manager\_name', and the 'department\_id' from the 'employees' table

SELECT first\_name || ' ' || last\_name AS Manager\_name, department\_id

-- Filtering rows based on the condition that the 'employee\_id' is in the result set of a subquery

FROM employees

-- Subquery to find 'manager\_id' values where the count of occurrences is greater than or equal to 4 in the 'employees' table

WHERE employee\_id IN

( SELECT manager\_id

FROM employees

-- Grouping 'manager\_id' values and filtering based on the count of occurrences

GROUP BY manager\_id

HAVING COUNT(\*) >= 4

);

**51.** From the following table, write a SQL query to find employees who have previously worked as 'Sales Representatives'. Return all the fields of jobs.

-- Selecting all columns (\*) from the 'jobs' table

SELECT \*

-- Filtering rows based on the condition that the 'job\_id' is in the result set of a nested subquery

FROM jobs

-- Subquery to find 'job\_id' values where the 'employee\_id' is in the result set of another subquery in the 'employees' table

WHERE job\_id IN

( SELECT job\_id

FROM employees

-- Subquery to find 'employee\_id' values where the 'job\_id' is 'SA\_REP' in the 'job\_history' table

WHERE employee\_id IN

( SELECT employee\_id

FROM job\_history

WHERE job\_id='SA\_REP'

)

);

**52.** From the following table, write a SQL query to find those employees who earn the second-lowest salary of all the employees. Return all the fields of employees.

-- Selecting all columns (\*) from the 'employees' table, aliased as 'm'

SELECT \*

-- Filtering rows from the 'employees' table, aliased as 'm', based on the condition that the count of distinct 'salary' values is equal to 2

FROM employees m

-- Subquery to count the number of distinct 'salary' values where the 'salary' is less than or equal to the 'salary' in the outer query

WHERE 2 = (SELECT COUNT(DISTINCT salary )

FROM employees

WHERE salary <= m.salary);

OR

SELECT m.\*

FROM employees m

JOIN employees e ON e.salary <= m.salary

GROUP BY m.employee\_id, m.salary

HAVING COUNT(DISTINCT e.salary) = 2;

**53.** From the following table, write a SQL query to find the departments managed by Susan. Return all the fields of departments.

-- Selecting all columns (\*) from the 'departments' table

SELECT \*

-- Filtering rows from the 'departments' table based on the condition that the 'manager\_id' is in the result set of a subquery

FROM departments

-- Subquery to find 'employee\_id' values where the 'first\_name' is 'Susan' in the 'employees' table

WHERE manager\_id IN

(SELECT employee\_id

FROM employees

WHERE first\_name='Susan');

**54.** From the following table, write a SQL query to find those employees who earn the highest salary in a department. Return department ID, employee name, and salary.

-- Selecting specific columns (department\_id, a.first\_name || ' ' || a.last\_name AS Employee\_name, salary) from the 'employees' table, aliased as 'a'

SELECT department\_id, first\_name || ' ' || last\_name AS Employee\_name, salary

-- Filtering rows from the 'employees' table, aliased as 'a', based on the condition that the 'salary' is equal to the maximum salary within the same 'department\_id'

FROM employees a

-- Subquery to find the maximum 'salary' where the 'department\_id' matches the 'department\_id' in the outer query

WHERE salary =

(SELECT MAX(salary)

FROM employees

WHERE department\_id = a.department\_id);

**55.** From the following table, write a SQL query to find those employees who have not had a job in the past. Return all the fields of employees.

-- Selecting all columns (\*) from the 'employees' table

SELECT \*

-- Filtering rows from the 'employees' table based on the condition that the 'employee\_id' is not in the result set of a subquery

FROM employees

-- Subquery to find 'employee\_id' values where there is no corresponding entry in the 'job\_history' table

WHERE employee\_id NOT IN

(SELECT employee\_id

FROM job\_history);